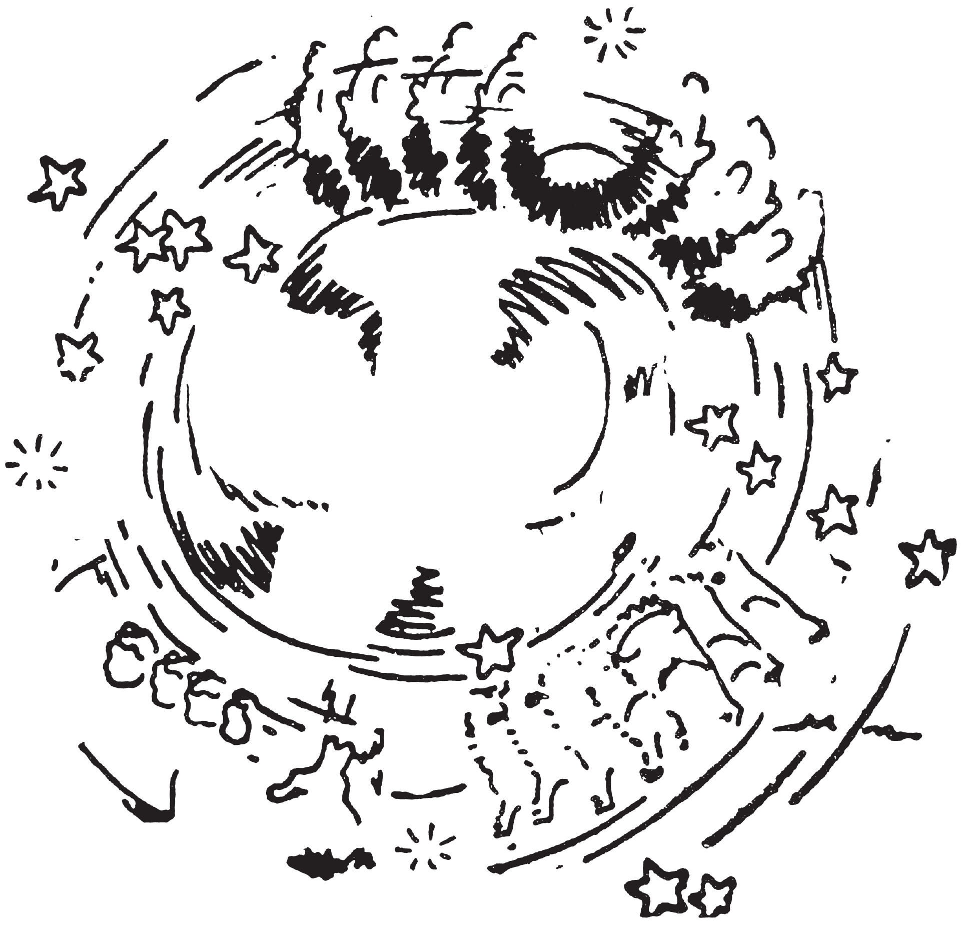


# Learning Local Abstractions in Complex Decision-Making Systems

# Learning Local Abstractions in Complex Decision-Making Systems

**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. H. Bijl,
chair of the Board for Doctorates
to be defended publicly on
Monday 30th, March 2026 at 12:30

by

**Elena CONGEDUTI**

This dissertation has been approved by the promotors.

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).

*Printed by:* Proefschriftmaken

*Cover by:* Andrea Salerno and Elena Congeduti

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

*To the first researcher I came to know*

# Contents

# SUMMARY

This thesis investigates how to learn local abstractions for scalable sequential decision making in large and complex systems. Real-world environments are typically dynamic, multiagent, and characterized by an extremely high number of state variables. As a result, exhaustive reasoning is computationally infeasible for an agent. Abstraction serves to reduce this complexity by focusing on the essential aspects of the environment while disregarding irrelevant details. A central theme of this work is the study of specific local abstractions and effective methods to learn them.

We introduce a perspective that unifies different approaches to state abstraction, showing how seemingly distinct methods can be systematically organized within a broader conceptual framework. This clarifies the connections between existing models and lays the foundation for more systematic development and reuse of abstraction techniques.

We explore approximate influence-based abstraction, an approach that enables building small local models complemented by learned representations of the external influence on local dynamics. We establish theoretical performance guarantees for approximate influence models, proving that the performance loss can be bounded in terms of the approximation error. The analysis is supported by empirical studies that show that accurate influence approximations improve performance in practice.

A further contribution is the investigation of the learnability of influence. We demonstrate that accurate influence representations can be learned efficiently, even in large-scale and long-horizon scenarios. Empirical evaluations show that small recurrent architectures are often sufficient to approximate the influence effectively and generalize beyond the training horizon.

In conclusion, this dissertation advances the foundations of state abstraction and principled applications of approximate influence-based abstraction. It provides a coherent framework for understanding state abstraction, establishes performance guarantees for approximate influence representations, and demonstrates the feasibility of influence learning. Together, these contributions offer insights into scalable and principled methods for sequential decision making in complex systems.

# SAMENVATTING

Dit proefschrift onderzoekt hoe lokale abstracties kunnen worden geleerd voor schaalbare sequentiële besluitvorming in grote en complexe systemen. Realistische omgevingen zijn doorgaans dynamisch, multi-agent en worden gekenmerkt door een extreem groot aantal toestandsvariabelen. Daardoor is uitputtend redeneren computationeel onuitvoerbaar voor een agent. Abstractie helpt deze complexiteit te reduceren door zich te richten op de essentiële aspecten van de omgeving en irrelevante details te negeren. Een centraal thema van dit werk is de studie van specifieke lokale abstracties en effectieve methoden om deze te leren.

Wij introduceren een perspectief dat verschillende benaderingen van toestandsabstractie verenigt en laat zien hoe ogenschijnlijk uiteenlopende methoden systematisch kunnen worden georganiseerd binnen een breder conceptueel kader. Dit verduidelijkt de verbanden tussen bestaande modellen en legt de basis voor een meer systematische ontwikkeling en hergebruik van abstractietechnieken.

Wij verkennen benaderde invloed-gebaseerde abstractie, een aanpak die het mogelijk maakt kleine lokale modellen te construeren die worden aangevuld met geleerde representaties van de externe invloed op lokale dynamiek. Wij formuleren theoretische prestatiegaranties voor benaderde invloedmodellen en tonen aan dat het prestatieniveau kan worden begrensd in termen van de benaderingsfout. Deze analyse wordt ondersteund door empirische studies die aantonen dat nauwkeurige invloedbenaderingen de prestaties in de praktijk verbeteren.

Een verdere bijdrage is het onderzoek naar de leerbaarheid van invloed. Wij tonen aan dat nauwkeurige invloedsrepresentaties efficiënt kunnen worden geleerd, zelfs in grootschalige en langetermijnscenario's. Empirische evaluaties laten zien dat kleine recurrente architecturen vaak voldoende zijn om de invloed effectief te benaderen en goed te generaliseren voorbij de trainingshorizon.

Concluderend versterkt dit proefschrift zowel de fundamenten van toestandsabstractie als de toepassing van benaderde invloed-gebaseerde abstractie. Het biedt een samenhangend raamwerk voor het begrijpen van toestandsabstractie, formuleert prestatiegaranties voor benaderde invloedsrepresentaties en demonstreert de haalbaarheid van invloedsleren. Gezamenlijk bieden deze bijdragen inzichten in schaalbare en principiële methoden voor sequentiële besluitvorming in complexe systemen.

# 1 Introduction

Understanding how to describe the influence of the external world when tackling a complex task is key to making smart and efficient decisions. In many real-world problems, the environment is large, dynamic, and composed of various interacting elements. Considering every detail in such settings can quickly become unmanageable. A common approach to this challenge is to rely on simplified representations of the environment. Some of these approaches adopt a local perspective, building models that focus on a restricted subset of the environment. While this strategy can significantly reduce the problem scale, it also risks neglecting critical information from the broader context. To address this, it is essential to understand how to efficiently capture external influence on local dynamics.

To illustrate these ideas, we consider a toy example. Suppose a commercial cleaning robot is responsible for keeping a space clean. The environment is dynamic: multiple people move around, at times reducing cleanliness in different areas. Many factors affect how waste accumulates, such as the arrangement of furniture, the location of entrances, and the presence of visitors. Moreover, when and how visitors generate waste depends on their needs, activities, interactions, and movements within the space. To determine which areas to prioritize and what cleaning actions to take, the robot must first address a fundamental question: how should it represent such a complex system? The answer to this question must account for two fundamental challenges the robot faces:

- Incomplete knowledge–the robot does not have precise information about the visitors' behavior and waste production patterns.
- Resource limitations–the robot needs to balance its computational and power resources to track relevant factors affecting cleanliness while ensuring it has enough capacity to develop an optimal cleaning plan.

The way the robot chooses to describe this system plays a crucial role in how effectively it can keep the room cleaned. Figure 1.1 illustrates a simple example scenario where a group of people gather in area B. If the robot neglects their presence, it might remain in area A or move elsewhere. Eventually, it will detect the accumulation of waste in B, rush over, and begin cleaning. However, this

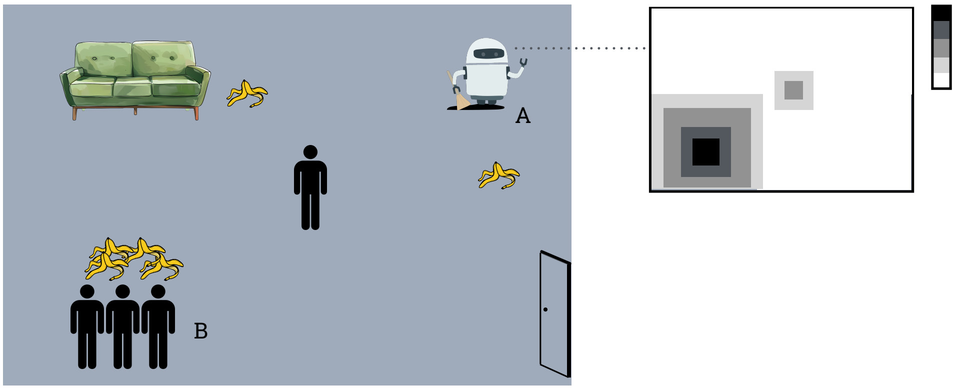


Figure 1.1.: A representation for the influence of the visitors on the cleaning robot problem as the density of people in the room.

means the space remains dirty for the time needed to identify the problem, travel to B, and complete the cleaning. Moreover, additional movements would consume battery power, reducing the total cleaning time before a recharge is needed. At the other extreme, the robot might attempt to track each individual. This approach would require reasoning over all visitors' decisions, interactions, and behaviors. Such a fine-grained level of description may be redundant and computationally unmanageable, preventing the robot from saving enough resources to timely develop a good strategy.

Suppose now that the robot recognizes that high-density areas are likely to quickly accumulate more waste. The spatial density is a more compact representation of the *influence* of the visitors on the cleaning robot decision-making problem that still retains sufficient information. In fact, knowing the distribution of the people allows the robot to invest its resources more thoroughly, focused on areas where higher presence of visitors is expected, and anticipate where cleaning will be needed. However, inferring or computing the density over time is typically computationally demanding even for small spaces. Instead, we take a different approach: learning to predict how the density evolves based on observations of the visitor behaviors.

In this dissertation, we analyze the relationship between different representation schemes, or *abstractions*, and we study a theoretical approach and empirical methods to represent the influence of the external world on one agent in the context of large and complex sequential decision-making problems.

## 1.1. SEQUENTIAL DECISION MAKING

Sequential decision making [1] refers to the problem of an agent making a sequence of decisions over time within a dynamic environment. An agent is any entity capable of processing information and selecting actions from a set of possible alternatives. The environment defines the causal rules that govern the world in which the agent operates, typically described by the evolution of a state. The state dynamics is typically described by probabilistic transitions. The agent interacts with the environment by receiving an observation and a scalar signal, a reward, as a consequence of its past decisions. The agent's goal is to find a decision making strategy, a policy, that maximizes the expected total reward collected over time, referred to as the value.

The study of sequential decision making has a long and rich history, dating back to early research in operations research and control theory [2]. As computers became more powerful and computation methods have been further developed, the scope of the sequential decision-making framework has been expanded to increasingly complex problems that do not require complete models or explicit knowledge of the environment, leading to the rise of reinforcement learning (RL) [3]. The integration of learning-based methods allowed for more flexible and scalable solutions and advances in deep reinforcement learning have further enabled breakthroughs in complex domains such as healthcare, autonomous systems, transportation systems, infrastructure management, and robotics [4, 5]. However, despite these successes, scaling these methods to real-world systems remains an open challenge. Two fundamental obstacles hinder scalability and adaptability to real-world applications. The first is the high dimensionality of the problem: describing accurately such problems typically requires an extremely large number of possible environmental states and actions. The second is the sample complexity: the number of interactions with the environment required to learn an effective policy is often prohibitively large. Fortunately, abstraction offers a promising approach to address these challenges.

## 1.2. ABSTRACTION

When making decisions, it is not feasible to consider all available information. Real-world events, scenarios, and environments are often extremely rich in details and thus inherently complex. However, typically only some aspects of reality are relevant for accomplishing a specific task. Rather than reasoning over an overwhelming volume of background knowledge, human learning and behavior are typically guided by structured knowledge representations. A body of work in cognitive science, neuroscience, and psychology has explored how abstract representations are formed and their role in human decision making [6–10]. Similarly, artificial agents face the challenge of dealing with complex observations of large, changing and noisy environments. Processing and understanding every detail can be both time and memory intensive. One strategy is to use abstraction to construct compressed and simplified representations that preserve only the information relevant to act effectively.

Some of the foundations of abstraction in artificial intelligence (AI), intended as the process of transforming the representation of a problem into a new (simplified) representation, were established within the field of theoretical artificial intelligence [11–15]. In particular, Giunchiglia and Walsh [16, 17] aimed to provide a unified

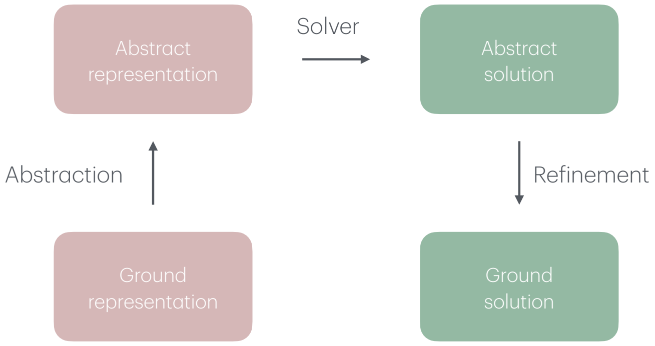


Figure 1.2.: The general abstraction workflow consists of constructing an abstract (simplified) representation of a ground problem, solving the abstract problem, and refining the abstract solution to apply it to the original setting.

framework for abstraction in reasoning, defined as a mapping from a *ground* problem to an *abstract* problem which is, in some sense, easier to solve. See Figure 1.2 for a schematic depiction of this process. After solving the abstract problem, an agent must refine the abstract solution to recover a policy that can be employed in the original ground-truth environment. The idea is that solving a problem through abstraction and refinement can be significantly more efficient than directly addressing the full complexity of the ground problem. Building on these foundational notions, the study of abstraction has progressively extended across various AI subfields and related disciplines, including planning [18–21], reinforcement learning [22–26], operational research [27], control theory [28, 29], and game theory [30].

In sequential decision making the performance of intelligent agents depends critically on how problems are represented. For instance, the level of detail and the choice of the features of the environment influence the agent's ability to learn smart strategies or execute effective plans. This reliance on representation is not surprising and is evident even in daily life. For example, people can easily navigate a city using a map that only shows the road network. Although this abstraction omits many details–such as contour lines, elevation, traffic lights, and road signs–it typically provides enough information to move around the city efficiently. However, identifying the essential aspects for many tasks is not straightforward. For instance, consider the cleaning robot example introduced in the chapter's introduction. The robot can model the presence of people in the room at different levels of granularity, ranging from neglecting them entirely to tracking each individual's position. Each option defines a distinct trade-off between computational efficiency and the accuracy of the representation. On the one hand, if the robot relies on an overly coarse representation, it risks overlooking the influence that the people have on waste production and may therefore plan its route suboptimally. However, reasoning about the position of single individuals may be infeasible and unnecessary. In essence, representation matters: selecting an appropriate abstraction is a critical modeling decision that directly influences one agent's ability to achieve its goal.

### 1.2.1. STATE ABSTRACTION

Abstraction, defined as the process of removing inessential information to focus on relevant aspects of a problem [31], is strictly related to the concept of aggregation. In the context of sequential decision making, aggregation typically refers to reducing the state space by grouping together situations that share similar characteristics. This process is commonly known as *state abstraction*. As an example, imagine that the cleaning robot moves around a building with many rooms. Instead of treating every location as a separate state, the agent can group all locations within the same room into single abstract states, such as "kitchen", "office", or "hallway". This aggregation map allows the agent to plan its route at the level of rooms rather than individual coordinates.

Other abstraction schemes are based on the aggregation of actions or behaviors, as in temporal abstraction [23, 32], where primitive one-step actions are aggregated to form temporally extended sequences called macro-actions or options. Hierarchical reinforcement learning [33–35] builds on this idea by organizing decision-making into multiple levels of hierarchical abstractions, where higher-level policies select among options rather than primitive actions. This approach naturally combines temporal abstraction with simplified state representations, as each level may operate over a different abstracted representation of the state space. Other lines of research in abstraction focus on modeling the uncertainty introduced by aggregation over states in different ways: function approximation [36–38] explores compact representations of the transition dynamics or value function, while game-based abstraction [39, 40] introduces a two-player game framework to model such uncertainty. Although these approaches rely on distinct core concepts or formal frameworks, they are strongly interrelated and are either grounded in or closely related to state abstraction.

Over the years, many state abstraction approaches and methods have been studied independently and in parallel across various research communities [21, 22, 39, 41–44]. As a result, a variety of formalisms, solution concepts, and notations have emerged. This fragmentation makes it difficult to compare methods directly, identify underlying connections, or transfer insights from one setting to another. One objective of this work is to bridge these gaps by analyzing the specific characteristics of different abstraction approaches and clarifying how they relate to each other. The ultimate goal is to provide a more coherent and unified foundation that supports the systematic development, comparison, and reuse of abstraction methods and results across research domains.

State abstraction methods often exploit specific similarities between states to construct (near) lossless abstractions. Specifically, performance guarantees typically rely on strong assumptions about the similarity of transitions or values among the aggregated states. Since these assumptions rarely hold in practice, such methods are often difficult to apply effectively [22, 45]. For instance, in the cleaning robot scenario, two locations within the same room may appear similar based on their proximity, yet differ substantially in their relevance for the decision-making task: a spot near a couch may accumulate more waste due to frequent visits, whereas a nearby empty area may remain comparatively clean. In such cases, state abstraction can either lead to overly coarse representations that degrade the quality of the policy or require preserving so many details that only a little reduction in problem complexity is achieved. While one could, in principle, formalize an objective that balances performance with computational cost, identifying such abstractions is generally intractable in practice. This practical limitation makes it difficult to guarantee a favorable trade-off between abstraction granularity and decision-making performance in real-world scenarios.

### 1.2.2. FACTORED LOCAL MODELS

Many large environments can be compactly represented using a factored structure, where the state is described in terms of a set of state variables or factors [46]. In the cleaning robot scenario, for example, the environment can be described by factors such as the robot's position, the location of waste, and the positions of visitors. A natural strategy for reducing the dimensionality of the state space is to abstract away entire factors, for instance, by ignoring the precise locations of individuals in the room. This corresponds to a special case of state abstraction, where states that coincide on the retained variables are grouped together. While ignoring entire state variables has the potential to exponentially reduce the state space size, it can also result in representations that discard relevant information. A body of work has investigated how to leverage such structural properties to construct state abstractions [47, 48]. However, most existing approaches rely on specific assumptions of independence among state variables or the irrelevance of certain factors with respect to the agent's task.

An alternative direction is to distinguish between factors that influence the decision-making problem more directly or locally, referred to as *local factors*, and to abstract away the others. This perspective is supported by research on localized abstractions [49–51], which is based on the observation that in many real-world domains, the rewards and observations are directly influenced by only a small subset of environmental variables. These approaches leverage such sparsity in the interaction structure to construct more compact local representations focused on locally relevant factors.

## 1.3. INFLUENCE REPRESENTATIONS

Influence-based abstraction (IBA) [52] builds on these previous local abstraction approaches. The key idea is to compensate for the loss of information by complementing the local abstraction with a representation of the influence that non-local variables exert on the local components. To illustrate this idea, consider the cleaning robot scenario. An abstract local model may include only the robot's position and the locations of waste, while excluding the positions of the visitors. In addition, the model incorporates a representation of the cumulative effect that people have on the waste distribution, such as the visitor density. This results in a much smaller and compact representation that still preserves the essential information needed to find effective cleaning policies.

IBA offers a flexible framework that generalizes prior local abstraction approaches without relying on strong assumptions such as independence or similarity between variables. A key advantage of this approach is that any abstract solution obtained through refinement corresponds to a valid ground solution. This property holds because the notion of *influence* fully captures the impact of the abstracted variables on the agent's local abstract model. Returning to the running example: if the cleaning robot can exactly track the population density, then individual trajectories become irrelevant for the task. However, deriving an analytic formulation or maintaining statistics of the density is often computationally demanding and may become unfeasible even for short-horizon problems. This also limits the resources the agent can allocate to the solution search. A more scalable alternative is to learn to predict the influence. This strategy is particularly promising in light of the availability of powerful deep learning methods for forecasting tasks [53–57].

The practical value of IBA crucially depends on whether it is actually feasible to learn accurate approximations of the influence in realistic settings. In particular, predicting the influence should be a substantially simpler and more sample-efficient task than solving the original control problem. In the cleaning robot scenario, predicting the density based on past observations of visitor behavior should turn out to be a relatively simple learning task. If visitors follow common motion patterns, a good approximation could be learned from a limited training sample, making this task considerably easier than reasoning about the future positions of each individual.

## 1.4. Research questions and contributions

This thesis focuses on the problem of learning representations of external influences on local abstractions, within the context of state abstraction for sequential decision making. As part of this investigation, we explore several approaches to state abstraction in order to convey a general and unified perspective on the topic. The contributions of this thesis are structured around the following research questions.

- What is a consistent perspective that generalizes standard state abstraction approaches?

  We investigate the structural connections among state abstraction methods developed within different research fields. By highlighting their formal relationships, we show that many seemingly distinct methods can be systematically organized within a broader scheme. This analysis naturally leads to the formulation of a unified conceptual framework that integrates the approaches examined (Chapter 2). Additionally, we provide a complete proof of the value bounds for a general state abstraction framework in the finite and infinite horizon cases and we show that such bounds are tight (Appendix 1).

- What performance guarantees can be established for influence approximations?

We study how to learn influence approximations while ensuring that the resulting performance loss remains bounded. Our theoretical and empirical results show that more accurate influence approximations lead to improved performance. Specifically, the performance loss can be bounded by a function of the approximation error. Unlike other state abstraction methods, this loss bound does not rely on structural similarities in transitions and rewards–assumption that usually fails to hold in realistic settings. Theoretical guarantees are aligned with empirical findings, which show that in practical scenarios, the performance loss is positively correlated with the approximation error (Chapter 3).

- To what extent and how can influence representations be efficiently learned in complex scenarios?

  We explore whether learning models can accurately approximate the influence in complex realistic settings, including large-scale systems and long-horizon problems. Our empirical analysis shows that this learning task is often manageable. We systematically compare different classes of learning models across a variety of scenarios to find that small recurrent neural network models can efficiently learn accurate influence approximations. A particular challenge arises in long-horizon scenarios, where collecting long training sequences becomes computationally expensive. To address this issue, we demonstrate that models trained on short sequences can generalize the influence effectively over longer time scales. Overall, these findings support the view that IBA offers a scalable and robust framework for decision making in complex, real-world environments (Chapter 4).

The following additional lines of work extend the core contributions of this thesis. They are not included in the main manuscript, as they do not fit well with the overall narrative.

In a preliminary investigation, we explore the problem of learning local models in real-world scenarios, in the specific context of a traffic forecasting application [58]. Specifically, we consider small individual intersections within a larger road network in the city of The Hague, The Netherlands. We train deep learning models on real-world traffic data to predict the local traffic flow. The results indicate that the learning models can successfully approximate local traffic dynamics. This work was led by my student Victoria Catalan Pastor, with my substantial contribution as a supervisor throughout the entire process.

Moreover, we investigate how to integrate the IBA approach into deep learning architectures for effective reinforcement learning [59, 60]. Our design combines a feedforward and a recurrent network to represent, respectively, the immediate effects of the observations on the action values and the temporal influences on local dynamics. This structure improves learning efficiency and mitigates convergence issues. In this case, I contributed to a project carried out by the entire research group, which was led by Miguel Suau De Castro. My specific role consisted in supporting the development of the experimental setup.

# REFERENCES

[1] M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[2] R. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[3] R. Sutton and A. Barto. *Reinforcement learning: an introduction*. MIT press, 2018.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, and G. Ostrovski. “Human-level control through deep reinforcement learning”. In: *Nature* (2015), p. 529.

[5] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* (2016), pp. 484–489.

[6] C. Kemp and J. Tenenbaum. “Structured statistical models of inductive reasoning.” In: *Psychological review* (2009), p. 20.

[7] J. Tenenbaum, C. Kemp, T. Griffiths, and N. Goodman. “How to grow a mind: Statistics, structure, and abstraction”. In: *Science* (2011), pp. 1279–1285.

[8] M. K. Ho, D. Abel, T. Griffiths, and M. Littman. “The value of abstraction”. In: *Current opinion in behavioral sciences* (2019), pp. 111–116.

[9] M. Botvinick, Y. Niv, and A. Barto. “Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective”. In: *Cognition* (2009), pp. 262–280.

[10] M. Eckstein and A. Collins. “Evidence for hierarchically-structured reinforcement learning in humans”. In: *Annual Meeting of the Cognitive Science Society*. Vol. 40. 2018.

[11] D. Plaisted. “Theorem proving with abstraction”. In: *Artificial Intelligence* (1981), pp. 47–108.

[12] J. Tenenberg. “Preserving consistency across abstraction mappings”. In: *International Joint Conference on Artificial (IJCAI)*. 1987.

[13] P. Nayak and A. Levy. “A semantic Theory of Abstractions”. In: *International Joint Conference on Artificial (IJCAI)*. 1995.

[14] S. De Saeger and A. Shimojima. “Channeling abstraction”. In: *International Symposium on Abstraction, Reformulation, and Approximation*. Springer. 2007.

[15] L. Saitta and J.-D. Zucker. *Abstraction in artificial intelligence and complex systems*. Springer, 2013.

[16] F. Giunchiglia and T. Walsh. “A theory of abstraction”. In: *Artificial Intelligence* (1992), pp. 323–389.

[17] F. Giunchiglia, A. Villafiorita, and T. Walsh. “Theories of abstraction”. In: *AI communications* (1997), pp. 167–176.

[18] C. Knoblock. “Automatically generating abstractions for planning”. In: *Artificial Intelligence* (1994), pp. 243–302.

[19] R. Dearden and C. Boutilier. “Abstraction and approximate decision-theoretic planning”. In: *Artificial Intelligence* (1997), pp. 219–283.

[20] C. Boutilier, T. Dean, and S. Hanks. “Decision-theoretic planning: structural assumptions and computational leverage”. In: *Journal of Artificial Intelligence Research (JAIR)* (1999), pp. 1–94.

[21] R. Givan, T. Dean, and M. Greig. “Equivalence notions and model minimization in Markov decision processes”. In: *Artificial Intelligence* (2003), pp. 163–223.

[22] A. K. McCallum. *Reinforcement learning with selective perception and hidden state*. University of Rochester, 1997.

[23] R. Sutton, D. Precup, and S. Singh. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial intelligence* (1999), pp. 181–211.

[24] L. Li, T. J. Walsh, and M. L. Littman. “Towards a unified theory of state abstraction for MDPs”. In: *International Symposium on Artificial Intelligence and Mathematics (AI&Math)*. 2006.

[25] M. Ponsen, M. Taylor, and K. Tuyls. “Abstraction and generalization in reinforcement learning: A summary and framework”. In: *International Workshop on Adaptive and Learning Agents*. Springer. 2009.

[26] D. Abel, D. Hershkowitz, and M. Littman. “Near optimal behavior via approximate state abstraction”. In: *International Conference on Machine Learning (ICML)*. 2016.

[27] D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans. “Aggregation and disaggregation techniques and methodology in optimization”. In: *Operations Research* (1991), pp. 553–582.

[28] B. Van Roy. “Performance loss bounds for approximate value iteration with state aggregation”. In: *Mathematics of Operations Research* (2006), pp. 234–244.

[29] D. P. Bertsekas, D. A. Castanon, *et al.* “Adaptive aggregation methods for infinite horizon dynamic programming”. In: (1988).

[30] D. Koller and A. Pfeffer. “Representations and solutions for game-theoretic problems”. In: *Artificial Intelligence* (1997), pp. 167–215.

[31] J. Kramer. “Is abstraction the key to computing?” In: *Communications of the ACM* (2007), pp. 36–42.

[32] M. Stolle and D. Precup. “Learning options in reinforcement learning”. In: *International Symposium on Abstraction, Reformulation and Approximation (SARA)*. Springer. 2002.

[33] A. Barto and S. Mahadevan. “Recent advances in hierarchical reinforcement learning”. In: *Discrete event dynamic systems* (2003), pp. 341–379.

[34] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier. “Hierarchical solution of Markov decision processes using macro-actions”. In: *Uncertainty in Artificial Intelligence (UAI)*. 1998.

[35] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek. “Hierarchical reinforcement learning: A comprehensive survey”. In: *ACM Computing Surveys (CSUR)* (2021), pp. 1–35.

[36] M. Hauskrecht. “Value-function approximations for partially observable Markov decision processes”. In: *Journal of artificial intelligence research* (2000), pp. 33–94.

[37] F. Melo, S. Meyn, and I. Ribeiro. “An analysis of reinforcement learning with function approximation”. In: *International Conference on Machine Learning (ICML)*. 2008.

[38] D. Bertsekas. *Abstract dynamic programming*. Athena Scientific, 2022.

[39] D. Parker, G. Norman, and M. Kwiatkowska. “Game-based abstraction for Markov decision processes”. In: *International Conference on the Quantitative Evaluation of Systems*. 2006, pp. 157–166.

[40] W. Wiesemann, D. Kuhn, and B. Rustem. “Robust Markov decision processes”. In: *Mathematics of Operations Research* (2013), pp. 153–183.

[41] T. Dean, R. Givan, and S. Leach. “Model reduction techniques for computing approximately optimal solutions for Markov decision processes”. In: *Uncertainty in Artificial Intelligence (UAI)*. 1997.

[42] B. Ravindran and A. Barto. “SMDP homomorphisms: an algebraic approach to abstraction in semi-Markov decision processes”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2003.

[43] R. Givan, S. Leach, and T. Dean. “Bounded-parameter Markov decision processes”. In: *Artificial Intelligence* (2000), pp. 71–109.

[44] M. Petrik and D. Subramanian. “RAAM: the benefits of robustness in approximating aggregated MDPs in reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[45] A. Bai, S. Srivastava, and S. Russell. “Markovian state and action abstractions for MDPs via hierarchical MCTS.” In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2016.

[46] C. Boutilier, R. Dearden, and M. Goldszmidt. “Stochastic dynamic programming with factored representations”. In: *Artificial Intelligence* (2000), pp. 49–107.

[47] N. Jong and P. Stone. “State abstraction discovery from irrelevant state variables”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2005.

[48] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. “Efficient solution algorithms for factored MDPs”. In: *Journal of Artificial Intelligence Research (JAIR)* (2003), pp. 399–468.

[49] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. “Solving transition independent decentralized Markov decision processes”. In: *Journal of Artificial Intelligence Research (JAIR)* (2004), pp. 423–455.

[50] P. Varakantham, J.-y. Kwak, M. E. Taylor, J. Marecki, P. Scerri, and M. Tambe. “Exploiting coordination locales in distributed POMDPs via social model shaping”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2009.

[51] S. Witwicki and E. Durfee. “Influence-based policy abstraction for weakly-coupled Dec-POMDPs”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2010.

[52] F. Oliehoek, S. Witwicki, and L. Kaelbling. “A sufficient statistic for influence in structured multiagent environments”. In: *Journal of Artificial Intelligence Research (JAIR)* (2021), pp. 789–870.

[53] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* (1997), pp. 1735–1780.

[54] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[55] S. Bai, Z. Kolter, and V. Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv preprint arXiv:1803.01271* (2018).

[56] F. Karim, S. Majumdar, H. Darabi, and S. Chen. “LSTM fully convolutional networks for time series classification”. In: *IEEE Access* (2017), pp. 1662–1669.

[57] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning phrase representations using RNN encoder–decoder for statistical machine translation”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[58] V. Catalán Pastor, E. Congeduti, and F. Oliehoek. “Overcoming traffic sensors malfunctions with deep learning”. In: *Benelux Conference on Artificial Intelligence (BNAIC) and Belgian Dutch Conference on Machine Learning (BeNeLearn)*. 2022.

[59] M. Suau, J. He, E. Congeduti, R. Starre, A. Czechowski, and F. Oliehoek. “Influence-aware memory architectures for deep reinforcement learning in POMDPs”. In: *Neural Computing and Applications* (2022), pp. 1–17.

[60] M. Suau, E. Congeduti, R. Starre, A. Czechowski, and F. Oliehoek. “Influence-based abstraction in deep reinforcement learning”. In: *AAMAS workshop on adaptive learning agents*. 2019.

# 2

# A CROSS-FIELD REVIEW OF STATE ABSTRACTION FOR MDPS

*Complex real-world systems pose a significant challenge to decision-making: an agent needs to explore a large environment, deal with incomplete or noisy information, generalize the experience, and learn from feedback to act optimally. These processes demand a great deal of representation capacity, thus putting a burden on the agent's limited computational and storage resources. State abstraction enables effective solutions by forming concise representations of the agents' world. As such, several research communities have extensively investigated this topic, and developed a variety of different abstraction approaches. Nevertheless, the relations among them still remain unseen or roughly defined. This hampers potential applications of solution methods whose scope remains limited to the specific abstraction context for which they have been designed. To this end, the goal of this paper is to organize the developed approaches and identify connections between abstraction schemes as a fundamental step towards methods generalization. As a second contribution, we discuss general abstraction properties with the aim of supporting a unified perspective for state abstraction.*

## 2.1. INTRODUCTION

Intelligent agents cannot reason about every detail of their large and structured world. They must necessarily base their decisions on a model of the environment that includes only a limited number of features. Intuitively, abstraction refers to the fundamental process of focusing on important aspects of the surroundings while ignoring irrelevant information. Through abstraction, an agent builds compressed representations of its environment retaining only the essential information for a specific task that can be used to solve more complex and large problems. As a

This work has been presented at 34th Benelux Conference on Artificial Intelligence (BNAIC) and the 30th Belgian Dutch Conference on Machine Learning (Benelearn) [1].

technique to ease decision making and learning algorithms in real-world scenarios and improve their scalability, abstraction has been extensively covered in the literature in artificial intelligence [2], operations research [3], theoretical computer science [4], and game theory [5].

In this work, we focus on abstraction for Markov decision processes (MDPs) [6], for which a variety of approaches have been proposed within different research fields ranging from game-based abstraction (GBA) [7], temporal abstraction [8], and value function approximation (VFA) [9]. Among them, state abstraction has been studied as a way to tackle computational issues and memory constraints when dealing with prohibitively large sizes of the state space. The key idea is to form aggregated MDPs whose *abstract states* correspond to clusters of the original *ground states*. Grouping states together necessarily introduces an additional degree of nondeterminism in the system due to the uncertainty about the actual ground state of the environment. In general, partitioning the state space results in non-Markov systems. Consequently, the induced stochastic process can not be modeled as an MDP straightforwardly. Several perspectives have been adopted to deal with this issue and have led to defining abstract MDPs or other structured processes potentially suited to approximate the stochastic process on the abstract space. One natural way to represent the uncertainty over the ground true states is by means of partial observability. In other words, the process over the abstract space can be seen as a partially observable Markov decision process (POMDP) [10] for a particular choice of the observation space and distribution [11, 12].

In this survey, we intend to highlight the close relationships among most of these approaches, showing that many coincide or are equivalent when viewed from the correct perspective. The goal of this investigation is to develop a theoretical understanding of these various approaches. Moreover, the relevance of a unified perspective lies in the possibility of leveraging techniques and solution methods from specific approaches in different abstraction contexts. The unifying perspective that emerges from these comparisons highlights the key role played by the history dependence in modeling the uncertainty on the ground state space. Finally, we present a general definition of state abstraction with the aim of conveying a more coherent perspective and support a unifying theoretical framework for state abstraction that can generalize most previous efforts in this direction. Based on this formal definition, we show how each abstraction scheme can be reinterpreted from this point of view. Despite recent attention that this topic has received, no unified theory has been established to date. The relationships between approaches remain unclear or loosely expressed, and potential limitations and advantages have not yet been thoroughly explored. In this regard, our work serves to bridge the gap between different research areas, with the aim of inspiring new methods and techniques from the cross-contamination of the different abstraction perspectives.

The rest of the paper is structured as follows. First, we provide an overview of previous work that surveys state abstraction. We introduce a high-level perspective on all abstraction approaches and suggest an integrated framework to classify them in Section 2.3. In Section 2.5, we introduce the formal definitions and discuss general abstraction properties. Section 2.5.1 formally describes the correspondence between state abstraction and POMDPs. Finally, in Section 2.4, we compare the different approaches and establish key connections.

### 2.1.1. Related surveys

The intuitive idea of abstraction as a map from one problem representation to a new one that preserves certain properties was introduced by Giunchiglia and Walsh [4]. Rogers *et al.* [3] describes different aggregation techniques to reduce the computational burden of solving large-scale optimization problems and discusses the corresponding error bounds. Although state abstraction for MDPs is directly addressed, the authors mainly focus on discussing different aggregation choices. Dearden and Boutilier [13] represents the first attempt to introduce a general framework for state abstraction using the propositional logic formalism. This work covers only cases where the aggregated states share the same dynamics but not the same rewards, thus partially addressing approximate abstraction.

The recent growing interest in reinforcement learning has given rise to several new abstraction approaches. Nonetheless, relatively few articles attempt to provide a comprehensive survey on the topic. Among them, Li, Walsh, and Littman [14] introduces a classification of abstraction approaches arranged hierarchically according to the level of information each approach preserves. The authors also propose a unified theoretical framework that generalizes many of the previous aggregation mechanisms, specifically those based on bisimulations [15, 16], MDP homomorphisms [17], utile distinction [12], and policy irrelevance [18]. However, several important approaches cannot be directly framed within their classification framework. Specifically, game-based abstraction [7], bounded parameter MDPs (BPMDPs) [19], and approaches based on robust control [20] are excluded. With our work, we intend to examine in detail the close interconnections that bind these approaches.

## 2.2. Background and notation

A Markov decision process is a tuple $\mathcal{M} = (S, A, T, R, \gamma)$ where $S$ and $A$ are finite state and action spaces, $T(s' \mid s, a)$ and $R(s, a)$ are the transition and reward functions and $\gamma \in (0, 1)$ the discount factor [6]. To interact with the environment, an agent employs a policy $\pi : S \longrightarrow A$ mapping states to actions. The objective of an agent is to maximize its expected cumulative reward obtained by executing a policy $\pi$, that is the value

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R^t \mid s, \pi\right]$$

when the process starts in state $s$, and $R^t = R(s^t, a^t)$ corresponds to the reward received at time step $t$ of the process. The target solution of the sequential decision problem modeled by the MDP, is the policy $\pi^*$ which maximizes the value $\pi^* = \operatorname{argmax}_{\pi} V^{\pi}$.

A partially observable Markov decision process is an MDP in which the agent is unaware of the actual state of the system. Instead, it receives partial information on the state of the environment through observations. Formally, a POMDP is a tuple $\mathscr{M}_{PO} = (S, A, O, T, \Omega, R, \gamma)$, where $(S, A, T, R, \gamma)$ describes an MDP, and $O$ and $\Omega(o \mid a, s')$ are the observation space and observation probabilities [10]. Given an action-observation history $h^t = (a^0, o^1, \ldots, a^{t-1}, o^t)$, the agent can keep track of a belief over the underlying state $b(s \mid h^t)$, representing the probability of being in state $s$ at time $t$ given the history $h^t$. A POMDP can be transformed into an MDP over the space of all the possible histories, i.e., *the belief space*, with rewards and deterministic transitions defined by

$$\begin{aligned} \rho(b, a) &= \sum_{s \in S} R(s, a) b(s \mid h^t) \\ b(s' \mid h^{t+1}) &= \frac{\Omega(o \mid a, s') \sum_{s \in S} T(s' \mid s, a) b(s \mid h^t)}{P(o \mid a, h^t)} \end{aligned} \tag{2.1}$$

for $h^{t+1} = (h^t, a, o)$. See Kaelbling, Littman, and Cassandra [10] for more details.

We mark all the abstract objects with an overline. For instance, an abstract state space is denoted as $\bar{S}$, and an abstract policy as $\bar{\pi} : \bar{S} \longrightarrow A$.

We use $\Delta(K)$ to represent the probability distribution simplex over the set $K$.

In general, we use capital letters to denote random variables and small letters for their instantiations.

## 2.3. ABSTRACTION CHOICES

Different modeling choices have led to the wide diversity in the state abstraction literature. Here, we provide an overview of the process of creating abstractions, highlighting the stages where different modeling decisions can be made and how various approaches arise accordingly. The idea is that identifying a general procedure and linking each approach to a specific stage of the abstraction process may help avoid common misconceptions, prevent inconsistent comparisons, and foster terminology alignment.

In the context of MDPs, abstraction refers to the process of mapping one MDP into a new representation that retains, to some extent, the Markov property. The general abstraction process can be thought of as the sequential application of the following steps:

1. Selection of the aggregation space.
2. Choice of the aggregation criteria.
3. Definition of the abstract dynamics.
4. Identification of a solution concept.

Although we do not assume that every approach has been developed strictly following this scheme, we believe that each can be characterized based on the authors' choices about these four stages.

### 2.3.1. AGGREGATION SPACE

The key advantage of abstraction lies in leveraging smaller representations to reduce the size of the initial problem, thereby easing the solution search. Common planning and reinforcement learning algorithms run in polynomial time in the size of the state space [21, 22], resulting in computationally intractable solutions for large state spaces of real-world scenarios. This naturally leads to aggregation as a method to define abstract spaces and raises the question: what is a suitable space to aggregate on? We distinguish between two types of approaches. The first class consists of approaches that aggregate over the state space, and in our work, we primarily focus on these. A second class performs aggregation at the level of action-state histories, such as feature MDPs [23] or influence-based abstraction (IBA) [24]. The distinction between these two classes is subtle: aggregation over histories is a type of state aggregation when trajectories are treated as states. Remarkably, aggregating over histories has the potential to overcome the loss of the Markov property. Just as action-observation histories $h^t$ in a POMDP exhibit a Markov dynamics, considering a space of histories increases the likelihood of finding aggregation schemes that preserve the Markov property of the system. On the other hand, the space of histories is exponentially larger than the state space.

### 2.3.2. AGGREGATION CRITERIA

Concretely, to define a state abstraction, we need to establish a partition over the state space. A practical way to induce such a partition is by using the preimage of an aggregation function.

**Definition 1.** An aggregation function for an MDP $\mathcal{M} = (S, A, T, R, \gamma)$ is a surjective function $\phi_{\bar{S}} : S \longrightarrow \bar{S}$ that maps each ground state $s$ to an abstract state $\bar{s} = \phi_{\bar{S}}(s)$, where generally $|\bar{S}| \ll |S|$.

With a slight abuse of notation, we use $\bar{s}$ to denote both the abstract state in $\bar{S}$ and the abstract class $\phi_{\bar{S}}^{-1}(\bar{s})$, i.e., the set of all ground states that $\phi_{\bar{S}}$ maps to $\bar{s}$. Moreover, we implicitly assume the context of a ground MDP $\mathcal{M} = (S, A, T, R, \gamma)$ and an aggregation function $\phi_{\bar{S}}$, unless otherwise stated.

Several aggregation criteria, or choices of the aggregation function, have been proposed based on different measures of similarity, such as stochastic bisimulations [15, 16], irrelevance criteria [14, 25], MDP homomorphisms [17], factors irrelevance [26]. Despite the diverse definitions, all these approaches consistently overlap. For instance, the irrelevance criteria generalize bisimulations, which, as defined by Givan, Dean, and Greig [15], are equivalent to the model irrelevance identified by Li, Walsh, and Littman [14]. MDPs homomorphisms also correspond to model irrelevance when the aggregation space is considered as the product of the state and action spaces $S \times A$. Factors irrelevance targets domains where the state space can be represented by state variables and abstracts away entire variables irrelevant to the model dynamics. This approach only induces exact abstraction, i.e., aggregation functions that cluster together states with identical transitions. We will discuss how the exact case of all approaches coincides for every choice of the aggregation function and model dynamics. Moreover, heuristic approaches, such as utile distinction [12] and policy irrelevance [18], have been considered. Methods for learning a good aggregation function, such as model reduction techniques [15] have also been introduced. We refer to Ferns *et al.* [27] for a complete survey of the metrics for state similarity in MDPs.

### 2.3.3. ABSTRACT DYNAMICS AND SOLUTIONS

Given an aggregation function, we can consider the stochastic process that an MDP naturally induces over the sets of aggregated states. In general, the aggregated process does not inherit the Markov property. The only exception is when states that have the same probability of reaching any abstract state are clustered together. In addition, if the reward function is preserved, then the stochastic process becomes an MDP, which we refer to as an exact abstraction.

**Definition 2** (Exact abstraction)**.** The aggregated process induced over $\bar{S}$ satisfies the Markov property if and only if for every $\bar{s} \in \bar{S}$ and $s_1, s_2 \in \bar{s}$

$$\sum_{s' \in \bar{s}'} T(s' \mid s_1, a) = \sum_{s' \in \bar{s}'} T(s' \mid s_2, a) \qquad \forall \bar{s}', \forall a \tag{2.2}$$

Moreover, if the reward function satisfies

$$R(s_1, a) = R(s_2, a) \qquad \forall a \tag{2.3}$$

then we call exact abstraction the MDP $\bar{\mathcal{M}} = (\bar{S}, A, \bar{T}, \bar{R}, \gamma)$ with transitions and rewards defined as

$$\bar{T}(\bar{s}' \mid \bar{s}, a) = \sum_{s' \in \bar{s}'} T(s' \mid s_1, a)$$
$$\bar{R}(\bar{s}, a) = R(s_1, a)$$

for any of the representative state $s_1 \in \bar{s}$.

Note that in general, an abstract policy $\bar{\pi} : \bar{S} \longrightarrow A$ can be naturally extended to a ground policy as $\bar{\pi}(s) \triangleq \bar{\pi}(\bar{s})$. In the exact case, the optimal abstract solution $\bar{\pi}^*$ for the abstract MDP $\bar{\mathcal{M}}$, when extended to a groun policy, coincides with the optimal solution for the underlying MDP [14]. We provide a complete and detailed proof in Appendix A.

However, in practice, states rarely behave exactly identically. As a result, exact abstractions achieve only a limited reduction of the state space. By relaxing the assumption on the similarities between grouped states, it is possible to induce a more substantial state space reduction, thereby significantly enhancing the efficiency of solution algorithms. For these *approximate* abstractions, however, the resulting abstract MDP is no longer equivalent to the original one. This implies that employing the optimal abstract policy $\bar{\pi}^*$ in the actual ground state space leads to a loss in the achieved value. We provide a detailed discussion on the bounds for the value loss associated with approximate state abstraction in Appendix A. Consequently, the problem becomes identifying a model consisting of model dynamics and a solution concept that well represents the aggregated process.

## 2.4. A COMPARISON OF APPROACHES

A primary distinction between representations lies in whether they identify a single abstract MDP or employ families of MDPs.

### 2.4.1. WEIGHTED AVERAGE ABSTRACTION

The first approach we discuss defines abstraction as a single MDP characterized by transition probabilities and rewards computed as weighted averages over the underlying states.

**Definition 3** (Weighted average abstraction [14]). Given a weighting function $\omega : S \longrightarrow [0,1]$ specifying a probability distribution over each abstract class, i.e., $\omega|_{\bar{s}} \in \Delta(\bar{s})$, a weighted average abstraction (WAA) is an MDP $\bar{\mathcal{M}}_\omega = (\bar{S}, A, \bar{T}_\omega, \bar{R}_\omega, \gamma)$ with transition probabilities and rewards defined as

$$\bar{T}_\omega(\bar{s}' \mid \bar{s}, a) = \sum_{s \in \bar{s}} \omega(s) \sum_{s' \in \bar{s}'} T(s' \mid s, a)$$
$$\bar{R}_\omega(\bar{s}, a) = \sum_{s \in \bar{s}} \omega(s)\, R(s, a)$$

Solving a WAA corresponds to finding the optimal policy $\bar{\pi}^*_\omega$ for the abstract MDP $\bar{\mathcal{M}}_\omega$.

Essentially, the weighting function serves to approximate the belief over the uncertain underlying ground states. As remarked by Bai, Srivastava, and Russell [11], the main limitation of this approach lies in modeling this uncertainty using a stationary weighting function independent of the process history. This may cause the abstract process to deviate significantly from the aggregated process. Consequently, the abstract optimal policy $\bar{\pi}^*_\omega$ may become completely ineffective. Without additional assumptions on the weighting function or the ground MDP dynamics, this solution corresponds to a myopic policy for a POMDP, which can incur an arbitrary value loss [28].

### 2.4.2. ABSTRACT BOUNDED PARAMETERS MDPS

By relaxing the assumption of approximating the belief with a constant function, alternative approaches address the uncertainty over ground states by considering families of MDPs. Bounded parameters MDPs generalize standard MDPs by replacing transition and reward function with real intervals. These intervals represent the minimal ranges required to include the ground transitions and rewards.

**Definition 4** (Abstract bounded parameters Markov decision processes [19]). An abstract bounded parameters MDP (ABPMDP) $(\bar{S}, A, \bar{T}_I, \bar{R}_I, \gamma)$ is defined by the following transitions and rewards

$$\bar{T}_I(\bar{s}' \mid \bar{s}, a) = \left[\min_{s \in \bar{s}} \sum_{s' \in \bar{s}'} T(s' \mid s, a),\ \max_{s \in \bar{s}} \sum_{s' \in \bar{s}'} T(s' \mid s, a)\right] \tag{2.4}$$

$$\bar{R}_I(\bar{s}, a) = \left[\min_{s \in \bar{s}} R(s, a),\ \max_{s \in \bar{s}} R(s, a)\right] \tag{2.5}$$

We consider the family of all abstract MDPs whose transitions and rewards lie in the intervals defined by (2.4) and (2.5)

$$\mathscr{F}_{\text{ABPMDP}} = \{\bar{\mathscr{M}} = (\bar{S}, A, \bar{T}, \bar{R}, \gamma)\, \bar{T}(\bar{s}' \mid \bar{s}, a) \in \bar{T}_I(\bar{s}' \mid \bar{s}, a),\, \bar{R}(\bar{s}, a) \in \bar{R}_I(\bar{s}, a)\ \forall \bar{s}, a, \bar{s}'\} \quad (2.6)$$

According to the definitions given in Section 2.2, the value $\bar{V}^{\bar{\pi}}_{\bar{\mathscr{M}}}$ corresponds to the expected discounted reward obtained by using a policy $\bar{\pi}$ in the abstract MDP $\bar{\mathscr{M}} \in \mathscr{F}$.

In this approach, two optimal policies are considered:

- The optimistically optimal policy $\bar{\pi}^*_{\text{opt}}$, which for every abstract policy $\bar{\pi}$ satisfies
$$\max_{\bar{\mathscr{M}} \in \mathscr{F}} \bar{V}^{\bar{\pi}}_{\bar{\mathscr{M}}} \leq \max_{\bar{\mathscr{M}} \in \mathscr{F}} \bar{V}^{\bar{\pi}^*_{\text{opt}}}_{\bar{\mathscr{M}}}$$
- The pessimistically optimal policy $\bar{\pi}^*_{\text{pes}}$, which for every abstract policy $\bar{\pi}$ satisfies
$$\min_{\bar{\mathscr{M}} \in \mathscr{F}} \bar{V}^{\bar{\pi}}_{\bar{\mathscr{M}}} \leq \min_{\bar{\mathscr{M}} \in \mathscr{F}} \bar{V}^{\bar{\pi}^*_{\text{pes}}}_{\bar{\mathscr{M}}}$$

In other words, the objective is to find the policies that perform optimally in the most and least favorable MDP within the ABPMDP family $\mathscr{F}$. Clearly, considering a general MDP from the ABPMDP family does not yield better results than considering a constant weighting function as in the previous approach. However, deriving solutions under pessimistic and optimistic assumptions allows to establish lower and upper bounds on optimal policies obtained in any abstract MDP within the ABPMDP family. We refer to Givan, Leach, and Dean [19] for the proof that these solution policies are well-defined and for further details on the bounds.

### 2.4.3. Abstract robust MDPs

Closely related to the concept of allowing uncertainties in transition and reward models, other abstraction formulations rely on game structures [7, 29]. We focus specifically on methods based on the robust MDP framework [30]. Robust MDPs originate from the idea of addressing the statistical uncertainty derived from estimating the system dynamics by introducing MDPs with imprecise transition probabilities [31]. Similarly to BPDMPS, this approach relaxes the definition of the transition and reward models. The key idea is to model the abstraction as a simple stochastic two-player game, where a second agent takes on the additional uncertainty introduced by the aggregation. We refer to this agent as the *nature agent*, as it chooses the environment by determining the transition and reward models. Specifically, at every time step, its action specifies a probability distribution over the current abstract state. The nature agent's action space corresponds to the ground state space $S$. Various representations of the nature state space are possible. We adopt the $(s,a)$-rectangularity assumption [20]. This means that the state space is represented as the product $\bar{S} \times A$ and the nature (stochastic) policy is defined by a function $\xi : \bar{S} \times A \longrightarrow \Delta(S)$, such that $\xi(\bar{s}, a)$ is a distribution over the abstract class $\bar{s}$, i.e., $\xi(\bar{s}, a) \in \Delta(\bar{s})$. Once the nature policy is set, the problem turns into a regular MDP.

**Definition 5** (Abstract robust Markov decision processes [20])**.** Given a nature policy $\xi$, an abstract robust MDP (ARMDP) is an MDP $\mathcal{M}_\xi = (\bar{S}, A, \bar{T}_\xi, \bar{R}_\xi, \gamma)$ where the transitions and rewards are defined as

$$\bar{T}_\xi(\bar{s}' \mid \bar{s}, a) = \sum_{s \in \bar{s}} \xi(\bar{s}, a)(s) \sum_{s' \in \bar{s}'} T(s' \mid s, a) \tag{2.7}$$

$$\bar{R}_\xi(\bar{s}, a) = \sum_{s \in \bar{s}} \xi(\bar{s}, a)(s)\ R(s, a) \tag{2.8}$$

Clearly, different choices for the nature policy lead to different MDPs. The solution $\bar{\pi}^*_\xi$ is defined assuming a fully adversarial behavior of the nature agent. Specifically, $\bar{\pi}^*_\xi$ attains the following maximum

$$\max_{\bar{\pi}} \min_{\xi} \bar{V}^{\bar{\pi}}_\xi(\bar{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \bar{R}^t_\xi \mid \bar{s}, \bar{\pi}\right]$$

### 2.4.4. Relationships

Each of the introduced approaches induces a set of abstract MDPs. Similar to the family of ABMDPs $\mathcal{F}_{\text{BPMDP}}$ defined by (2.6), we consider the following sets

$$\mathcal{F}_{\text{WAA}} = \{\bar{\mathcal{M}}_\omega : \omega \in \Delta(S), \omega|_{\bar{s}} \in \Delta(\bar{s})\ \forall \bar{s}\} \tag{2.9}$$

$$\mathcal{F}_{\text{ARMDP}} = \{\bar{\mathcal{M}}_\xi : \xi(\bar{s}, a) \in \Delta(S),\ \xi(\bar{s}, a)|_{\bar{s}} \in \Delta(\bar{s})\ \forall \bar{s}, a\} \tag{2.10}$$

where $\omega|_{\bar{s}}$ and $\xi(\bar{s}, a)|_{\bar{s}}$ denote the function restrictions to the states $s \in \bar{s}$. Equations (2.9) and (2.10) define, respectively, the family of all possible weighted average abstractions and the family of abstract MDPs for every possible nature policy. The relationships connecting all these approaches follow directly from the definitions. The following statement presents the main result of our work and aims to delineate these relationships.

**Theorem 1.** Consider the WAA, ABPMDP, and ARMDP as in Definitions 3, 4, 5, along with the corresponding families of abstract MDPs $\mathcal{F}_{\text{WAA}}$, $\mathcal{F}_{\text{ABPMDP}}$, $\mathcal{F}_{\text{ARMDP}}$ as defined in (2.9), (2.6),(2.10). Then, the following claims hold:

1. The family of weighted average abstractions is a subset of the abstract MDPs generated by nature policies, which, in turn, is a subset of the abstract BPMDP family. Formally, the following chain of inclusions holds

$$\mathcal{F}_{\text{WAA}} \subseteq \mathcal{F}_{\text{ARMDP}} \subseteq \mathcal{F}_{\text{ABPMDP}}$$

2. If the aggregation function defines an exact abstraction, i.e., (2.2) and (2.3) hold, then

$$\mathcal{F}_{\text{WAA}} = \mathcal{F}_{\text{ARMDP}} = \mathcal{F}_{\text{ABPMDP}}$$

In this case, all three sets degenerate to the same abstract MDP corresponding to the exact abstraction described in Definition 2.

*Proof.* 1. To show the first inclusion, we consider an arbitrary weighting function $\omega$ and the MDP $\bar{\mathcal{M}}_\omega \in \mathcal{F}_{\mathrm{WAA}}$. According to (2.7), the nature policy $\xi(\bar{s}, a) := \omega|_{\bar{s}}$ induces an equivalent abstract robust MDP $\bar{\mathcal{M}}_\xi = \bar{\mathcal{M}}_\omega \in \mathcal{F}_{\mathrm{ARMDP}}$. In essence, the subset of nature policies $\xi$ that are independent of the actions, $\xi(\bar{s}, a_1) = \xi(\bar{s}, a_2)$ for any $a_1, a_2 \in A$, spans the entire family $\mathcal{F}_{\mathrm{WAA}}$. In order to show the second inclusion, it suffices to observe that for any policy $\xi$ the transitions and rewards defined by (2.7) satisfy

$$\begin{aligned}\bar{T}_\xi(\bar{s}' \mid \bar{s}, a) &\leq \sum_{s \in \bar{s}} \xi(\bar{s}, a)(s) \max_{s \in \bar{s}} \left( \sum_{s' \in \bar{s}'} T(s' \mid s, a) \right) \\ &= \max_{s \in \bar{s}} \left( \sum_{s' \in \bar{s}'} T(s' \mid s, a) \right) \sum_{s \in \bar{s}} \xi(\bar{s}, a)(s) \\ &= \max_{s \in \bar{s}} \sum_{s' \in \bar{s}'} T(s' \mid s, a)\end{aligned}$$

Likewise, by considering the minimum, we can show that $\bar{T}_\xi(\bar{s}' \mid \bar{s}, a) \geq \min_{s \in \bar{s}} \sum_{s' \in \bar{s}'} T(s' \mid s, a)$, which proves that $\bar{T}_\xi(\bar{s}' \mid \bar{s}, a) \in \bar{T}_I(\bar{s}' \mid \bar{s}, a)$. Similarly, it follows that $\bar{R}_\xi(\bar{s}, a) \in \bar{R}_I(\bar{s}, a)$.

2. First, we demonstrate that the intervals in (2.4) and (2.5), which define the abstract family $\mathcal{F}_{\mathrm{ABPMDP}}$, reduce to a single point. Specifically, from the assumptions (2.2) and (2.3), we have

$$\begin{aligned}\bar{T}_I(\bar{s}' \mid \bar{s}, a) &= \left[ \sum_{s' \in \bar{s}'} T(s' \mid s_1, a)\,,\ \sum_{s' \in \bar{s}'} T(s' \mid s_1, a) \right] = T(s' \mid s_1, a) \\ \bar{R}_I(\bar{s}, a) &= [R(s_1, a)\,,\ R(s_1, a)] = R(s_1, a)\end{aligned}$$

for an arbitrary choice of $s_1 \in \bar{s}$. Therefore, $\mathcal{F}_{\mathrm{BPMDP}}$ contains only the abstract exact MDP. The equality then follows directly from the inclusions established in step 1.

$\square$

According to the second claim of Theorem 1, when complete equivalence between aggregated states holds, all approaches yield an exact approximation of the aggregated process, which naturally inherits the Markov property. In the general case, the families do not coincide, and thus neither do their solutions. Nonetheless, a weighted average abstraction corresponds to a nature policy independent of the agent actions. In turn, for each nature policy, an abstract robust MDP can be interpreted as an MDP within the ABPMDP family. These inclusions are determined by the different dependencies of the uncertainty model: a weighing function is a history-independent function; the nature policy, assuming the $(s, a)$-rectangularity structure, depends on the current action; and the ABPMDP introduces an additional dependence on the next abstract state.

Another class of state abstraction approaches relies on value function approximations (VFA) [9, 32]. The central idea is to approximate the non-Markovian model dynamics by employing projections onto the aggregated space. Scpecifically, the projections capture the uncertainty over the state space, as each ground state is mapped onto the abstract space through a predefined projection function. Depending on chosen projection metric, different abstract MDPs can be defined. In this sense, the metrics serve the same role as weighting functions, making the two approaches completely equivalent. Although we do not discuss explicitly this case in Theorem 1, formalizing the equivalence is straightforward. It suffices to observe that the structure of the projections used depends solely on the current abstract state and coincides precisely with that of a weighting function.

Table 2.1 summarizes the main features of all the approaches covered in this survey. The overall picture that emerges consists of abstraction schemes which deal with uncertainty by introducing families of abstract MDPs and deriving solution concepts by targeting a specific MDP within the family. The key distinguishing factor is the extent to which the abstract state-action history determines the uncertainty model. In the next section, we explore the relevance of including the abstract trajectories in the uncertainty model and how, from this perspective, abstraction can be interpreted as a form of partial observability.

| **Approach** | **Aggregation Space** | **Abstract representation** | **Solution Concept** | **Type** | **Uncertainty** |
|---|---|---|---|---|---|
| IBA [24] | histories | MDP | optimality | exact | - |
| ABPMDPs [19] | states | MDP set | pessimism /optimism | approx | $\lambda(\bar{s}, a, \bar{s}')$ |
| ARMDPs [20] | states | Robust MDP | pessimism | approx | $\lambda(\bar{s}, a)$ |
| WAA [14] | states | MDP | optimality | approx | $\lambda(\bar{s})$ |
| GBA[7] | states | stochastic game | pessimism /optimism | approx | - |
| VFA [9] | states | MDP | optimality | approx | $\lambda(\bar{s})$ |
| POMDPs [11] | states | POMDP | optimality | approx | $\lambda(\bar{h}^t)$ |

Table 2.1.: Schematic representation of abstraction approaches.

## 2.5. A GENERAL FRAMEWORK

The analysis of similarities among state abstraction approaches leads to a universal definition which encompasses them all.

We define the sets of candidate uncertainty functions as

$$\begin{aligned} \mathcal{U}_T &= \left\{\lambda_T(\bar{s}, a, \bar{s}') : \lambda_T(\bar{s}, a, \bar{s}') \in \Delta(\bar{s}) \ \forall \bar{s}, a, \bar{s}'\right\} \\ \mathcal{U}_R &= \{\lambda_R(\bar{s}, a) : \lambda_R(\bar{s}, a) \in \Delta(\bar{s}) \ \forall \bar{s}, a\} \end{aligned} \tag{2.11}$$

Each pair $\lambda = (\lambda_T, \lambda_R) \in \mathcal{U}_T \times \mathcal{U}_R$ uniquely defines an abstract MDP $\bar{\mathcal{M}}_\lambda = (\bar{S}, A, \bar{\mathcal{T}}_\lambda, \bar{R}_\lambda, \gamma)$ with transitions and rewards given by

$$\begin{aligned} \bar{T}_\lambda(\bar{s}' \mid \bar{s}, a) &= \sum_{s \in \bar{s}} \lambda_T(\bar{s}, a, \bar{s}')(s) \sum_{s' \in \bar{s}'} T(s' \mid s, a) \\ \bar{R}_\lambda(\bar{s}, a) &= \sum_{s \in \bar{s}} \lambda_R(\bar{s}, a)(s) R(s, a) \end{aligned}$$

**Definition 6** (State abstraction)**.** Given an uncertainty set $\mathcal{U} \subset \mathcal{U}_T \times \mathcal{U}_R$, a state abstraction is the family of MDPs

$$\mathcal{F}_{\mathcal{U}} = \{\bar{\mathcal{M}}_\lambda = (\bar{S}, A, \bar{T}_\lambda, \bar{R}_\lambda, \gamma) : \lambda \in \mathcal{U}\}$$

The uncertainty set $\mathcal{U}$ encodes the additional nondeterminism introduced by aggregation, and its characterizations capture the differences between abstraction approaches. The following proposition shows how different choices of the uncertainty set $\mathcal{U}$ correspond to the abstraction approaches described in the previous section.

**Proposition 1.** Consider an uncertainty set $\mathcal{U} \subset \mathcal{U}_T \times \mathcal{U}_R$ and the state abstraction $\mathcal{F}_{\mathcal{U}}$ according to Definition 6.

- **Weighted average abstraction:** if the uncertainty set is independent of both the actions and next abstract states, i.e., $\mathcal{U} = \{(\lambda_T, \lambda_R) : \lambda_T = \lambda_R = \lambda(\bar{s}) \in \Delta(\bar{s})\}$, then the state abstraction corresponds to the weighted average abstractions family,
$$\mathcal{F}_{\mathcal{U}} = \mathcal{F}_{\text{WAA}}$$
- **Abstract robust MDP:** if the uncertainty set depends on the actions but remains independent of the next abstract states, i.e., $\mathcal{U} = \{(\lambda_T, \lambda_R) : \lambda_T = \lambda_R = \lambda(\bar{s}, a) \in \Delta(\bar{s})\}$, then the state abstraction corresponds to the abstract robust MDP family,
$$\mathcal{F}_{\mathcal{U}} = \mathcal{F}_{\text{ARMDP}}$$
- **Abstract bounded parameters MDP:** if the uncertainty set depends on both the actions and on the next abstract states, i.e., $\mathcal{U} = \mathcal{U}_T \times \mathcal{U}_R$, then the state abstraction corresponds to the abstract bounded parameter MDP
$$\mathcal{F}_{\mathcal{U}} = \mathcal{F}_{\text{ABPMDP}}$$

### 2.5.1. ABSTRACTION AS PARTIAL OBSERVABILITY

Bai, Srivastava, and Russell [11] shows that the stochastic process over the aggregated space corresponds to a POMDP, with the original ground MDP as the underlying MDP and the abstract states as observations. The key insight is that the aggregated process generally cannot be well approximated by Markov dynamics. Instead, by including histories in the state space, it become possible to define the abstract process as a POMDP.

**Definition 7** (Abstract POMDP)**.** An abstract POMDP $\bar{\mathcal{M}}_{\text{POMDP}} = (S, A, \bar{S}, T, \Omega, , R, \gamma)$ is a POMDP with $(S, A, T, R, \gamma)$ as underlying MDP. The observation space consists of the abstract space $O = \bar{S}$ with deterministic observation probabilities defined as

$$\Omega(\bar{s}' \mid a, s') = \mathbb{1}_{s' \in \bar{s}'}$$

To highlight the relationship between the abstract POMDP and Definition 6, we extend the notion of uncertainty to include the entire action-abstract state history

$\bar{h}^t = (\bar{s}^0, a^1, \ldots, a^{t-1}, \bar{s}^t)$, by setting $\mathscr{U} = \{\lambda_T = \lambda_R = \lambda(\bar{h}^t) \in \Delta(\bar{s}^t)\}$. Then given the transitions and rewards

$$\bar{T}_\lambda(\bar{s}^{t+1} \mid \bar{h}^t, a^t) = \sum_{s^t \in \bar{s}^t} \lambda(\bar{h}^t)(s^t) \sum_{s^{t+1} \in \bar{s}^{t+1}} T(s^{t+1} \mid s^t, a^t)$$
$$\bar{R}_\lambda(\bar{h}^t, a^t) = \sum_{s^t \in \bar{s}^t} \lambda(\bar{h}^t)(s) R(s^t, a^t)$$

The family $\mathscr{F}_\mathscr{U}$ also includes non-Markov processes. If we impose the additional constraint that the lambda functions follow the belief update rule expressed by (2.1), resulting in $\mathscr{U} = \{\lambda_T = \lambda_R = \lambda(\bar{h}^t) \in \Delta(\bar{s}^t), \lambda(\bar{h}^t) = \text{belief-up}(\bar{h}^{t-1}, a^t, \bar{s}^t)\}$, then the state abstraction $\mathscr{F}_\mathscr{U}$ is equivalent to the abstract POMDP $\bar{\mathscr{M}}_{\text{POMDP}}$ and provides an exact description of the aggregated process.

This argument highlights the necessity of incorporating history when modeling uncertainty for state abstraction. For example, in the WAA case, the belief $\lambda(\bar{h}^t)$ is approximated by an arbitrary history-independent function $\omega = \lambda(\bar{s}^t)$. Consequently, we can achieve at most the value of an optimal memoryless policy, which is generally known to yield arbitrarily poor performance [33]. On the other hand, one may argue that modeling state abstraction as a POMDP would compromise the primary benefit of abstraction–reducing the problem size [12]. A possible approach is to adopt intermediate solutions that partially incorporate the history and explore trade-offs between problem size reduction and history dependency.

## 2.6. CONCLUSIONS

In this paper, we survey the main approaches to state abstraction developed within reinforcement learning, planning, operations research, and game theory communities. We characterize a general abstraction scheme by identifying stages of the abstraction-building process corresponding to the choice of the aggregation space and function, the model dynamics and solution concept. Our analysis primarily focused on comparing techniques introduced to model the abstract dynamics and solutions as weighted average abstractions, bounded parameters MDPs, robust MDPs and value function approximations. We show how they can all be interpreted under the unified perspective of families of abstract MDPs, where the distinctions arise from the employed model of the uncertainty. Finally, we introduce a unified formal framework that generalizes all prior approaches and highlight how the partial observability perspective can be embedded into this general definition of state abstraction.

# REFERENCES

[1] E. Congeduti and O. Frans. “A cross-field review of state abstraction for Markov decision processes”. In: *Benelux Conference on Artificial Intelligence (BNAIC) and Belgian Dutch Conference on Machine Learning (BeNeLearn)*. 2022.

[2] L. Saitta and J.-D. Zucker. *Abstraction in artificial intelligence and complex systems*. Springer, 2013.

[3] D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans. “Aggregation and disaggregation techniques and methodology in optimization”. In: *Operations Research* (1991), pp. 553–582.

[4] F. Giunchiglia and T. Walsh. “A theory of abstraction”. In: *Artificial Intelligence* (1992), pp. 323–389.

[5] D. Koller and A. Pfeffer. “Representations and solutions for game-theoretic problems”. In: *Artificial Intelligence* (1997), pp. 167–215.

[6] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[7] D. Parker, G. Norman, and M. Kwiatkowska. “Game-based abstraction for Markov decision processes”. In: *International Conference on the Quantitative Evaluation of Systems*. 2006, pp. 157–166.

[8] R. Sutton, D. Precup, and S. Singh. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial intelligence* (1999), pp. 181–211.

[9] M. G. Lagoudakis. *Value Function Approximation*. Springer, 2017.

[10] L. Kaelbling, M. Littman, and A. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* (1998), pp. 99–134.

[11] A. Bai, S. Srivastava, and S. Russell. “Markovian state and action abstractions for MDPs via hierarchical MCTS.” In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2016.

[12] A. K. McCallum. *Reinforcement learning with selective perception and hidden state*. University of Rochester, 1997.

[13] R. Dearden and C. Boutilier. “Abstraction and approximate decision-theoretic planning”. In: *Artificial Intelligence* (1997), pp. 219–283.

[14] L. Li, T. J. Walsh, and M. L. Littman. “Towards a unified theory of state abstraction for MDPs”. In: *International Symposium on Artificial Intelligence and Mathematics (AI&Math)*. 2006.

[15] R. Givan, T. Dean, and M. Greig. “Equivalence notions and model minimization in Markov decision processes”. In: *Artificial Intelligence* (2003), pp. 163–223.

[16] T. Dean, R. Givan, and S. Leach. “Model reduction techniques for computing approximately optimal solutions for Markov decision processes”. In: *Uncertainty in Artificial Intelligence (UAI)*. 1997.

[17] B. Ravindran and A. Barto. “SMDP homomorphisms: an algebraic approach to abstraction in semi-Markov decision processes”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2003.

[18] N. Jong and P. Stone. “State abstraction discovery from irrelevant state variables”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2005.

[19] R. Givan, S. Leach, and T. Dean. “Bounded-parameter Markov decision processes”. In: *Artificial Intelligence* (2000), pp. 71–109.

[20] M. Petrik and D. Subramanian. “RAAM: the benefits of robustness in approximating aggregated MDPs in reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[21] A. Strehl, L. Li, and M. Littman. “Reinforcement learning in finite MDPs: PAC analysis.” In: *Journal of Machine Learning Research (JMLR)* (2009).

[22] M. Littman, T. Dean, and L. P. Kaelbling. “On the complexity of solving Markov decision problems”. In: *Uncertainty in Artificial Intelligence (UAI)*. 1995.

[23] M. Hutter. “Extreme state aggregation beyond MDPs”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2014, pp. 185–199.

[24] F. Oliehoek, S. Witwicki, and L. Kaelbling. “A sufficient statistic for influence in structured multiagent environments”. In: *Journal of Artificial Intelligence Research (JAIR)* (2021), pp. 789–870.

[25] D. Abel, D. Hershkowitz, and M. Littman. “Near optimal behavior via approximate state abstraction”. In: *International Conference on Machine Learning (ICML)*. 2016.

[26] C. Boutilier, R. Dearden, and M. Goldszmidt. “Stochastic dynamic programming with factored representations”. In: *Artificial Intelligence* (2000), pp. 49–107.

[27] N. Ferns, P. Castro, D. Precup, and P. Panangaden. “Methods for computing state similarity in Markov decision processes”. In: *Uncertainty in Artificial Intelligence (UAI)*. 2006.

[28] S. P. Singh and R. C. Yee. “An upper bound on the loss from approximate optimal-value functions”. In: *Machine Learning* (1994), pp. 227–233.

[29] L. Winterer, S. Junges, R. Wimmer, N. Jansen, U. Topcu, J.-P. Katoen, and B. Becker. “Motion planning under partial observability using game-based abstraction”. In: *IEEE Conference on Decision and Control (CDC)*. 2017, pp. 2201–2208.

[30] W. Wiesemann, D. Kuhn, and B. Rustem. “Robust Markov decision processes”. In: *Mathematics of Operations Research* (2013), pp. 153–183.

[31] C. C. White III and H. K. Eldeib. “Markov decision processes with imprecise transition probabilities”. In: *Mathematics of Operations Research* (1994), pp. 739–749.

[32] B. Van Roy. “Performance loss bounds for approximate value iteration with state aggregation”. In: *Mathematics of Operations Research* (2006), pp. 234–244.

[33] M. Littman. “Memoryless policies: theoretical limitations and practical results”. In: *International Conference on Simulation of Adaptive Behavior*. 1994.

# 3
# Loss bounds for approximate influence-based abstraction

*Sequential decision-making techniques hold great promise to improve the performance of many real-world systems, but computational complexity hampers their principled application. Influence-based abstraction aims to gain leverage by modeling local subproblems together with the 'influence' that the rest of the system exerts on them. While computing exact representations of such influence might be intractable, learning approximate representations offers a promising approach to enable scalable solutions. This paper investigates the performance of such approaches from a theoretical perspective. The primary contribution is the derivation of sufficient conditions on approximate influence representations that can guarantee solutions with small value loss. In particular we show that neural networks trained with cross entropy are well suited to learn approximate influence representations. Moreover, we provide a sample based formulation of the bounds, which reduces the gap to applications. Finally, driven by our theoretical insights, we propose approximation error estimators, which empirically reveal to correlate well with the value loss.*

## 3.1. Introduction

Sequential decision-making methods have the potential to improve control in distributed or networked systems that can involve many agents and complex environments. However, applying these methods in a principled manner is often difficult due to computational complexity. One idea to improve scalability is using abstractions, compressed representations of the sufficient information for an agent to perform optimal decisions. Many abstraction approaches have been suggested [2–4] but in order to ensure small value loss, they only allow abstracting states of the environment with similar dynamics. A body of work on localized abstractions [5–8]

This chapter has been published in the proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), (2021) [1].

tries to overcome this issue, allowing to abstract a large part of the system away. These approaches leverage the sparse interaction structures of many real-world domains where the agent's reward and observations depend directly only on a few (local) state variables.

*Influence-based abstraction* (IBA) [9] generalizes over these approaches and provides a unified framework for localized abstractions for a general class of multiagent problems. The basic idea is to decompose structured multiagent systems into small submodels for each agent where only few local variables are included. The 'influence' summarizes the indirect effects of other agents policies and the rest of the environment on the local variables of a single agent. Unlike existing methods to compute best responses [10–12] which involve reasoning over the entire state space and other agent's action-observation histories, IBA ensures that the decision maker needs to reason only over few local state variables. As a result, the best-response problem can be solved much more efficiently in the local model without any loss in value. This approach may not only enable speedups in best-response problems but may serve also as the foundation for scalable multiagent approaches based, for instance, on searching in the space of influences [5].

However, even though IBA provides a lossless abstraction, in general, deducing an *exact representation* of the influence itself is computationally intractable even for simple problems. Given the rapid progress in learning methods for sequence prediction [13–15], we envision that inducing *approximate representations* of the influence offers a promising approach to scalable solutions. This raises a fundamental question: to what extent are such existing methods (and their proxy-loss functions) aligned with the actual influence-base abstraction objective of achieving near-optimal value?

In this work, we address this issue, showing that there is theoretical and empirical evidence that employing existing sequence predictors can lead to good approximate influence representations. In particular, we derive a performance loss bound as well as a probabilistic version that provides quality guarantees on approximate influence to ensure near-optimal solutions. Like bounds based on the Bellman error, our bounds themselves are computationally intractable for large problems. This is to be expected: the performance loss is characterized by the quality of each possible prediction. Therefore, deriving an exact bound essentially corresponds to computing the exact generalization error of the machine learning model employed. Nevertheless, we show that a neural network trained with cross-entropy loss seems well suited for the prediction task, as the training objective is aligned with the bound we derived. Finally, we empirically demonstrate that it is possible to compute statistics, based on the empirical test error of approximate influence, that correlate well with the actual value loss.

In summary, our contributions in this paper are:

1. Sufficient conditions for the quality of the influence approximation in terms of value loss.
2. Discussing how these conditions are aligned with usual optimization of neural networks trained with cross entropy loss.
3. An empirical evaluation showing that optimizing cross entropy loss of influence

predictors leads to lower performance loss.

The paper is organized as follows. We elaborate on related work in Section 3.2. In Section 3.3, we outline the background on IBA for sequential decision-making problems. Then, we explain the general procedure to exploit approximate-influence representations in planning problems in Section 3.4. In Section 3.5, we introduce the value loss bounds derived and we discuss their implications for multiagent systems in Section 3.6. Section 3.7 presents the empirical evaluations. Finally, we discuss future work and conclusions in Section 3.8.

## 3.2. Related work

Abstraction has been widely studied as a technique to accelerate learning and improve scalability in complex (multiagent) problems. Many abstraction approaches have been suggested [2, 3, 16–19] but they all share a common limitation: in order to guarantee bounded loss, the ground states in the same abstract states (which are clusters of ground states) must all share similar transition probabilities. However, when we want to abstract away entire sets of state variables, such guarantees will typically not be possible. Influence-based abstraction differs essentially from these kind of abstractions. In fact, it enables abstracting entire factors still providing guarantees of optimality when the aggregated states have typically very different transition probabilities.

## 3.3. Background

First we outline the background on sequential decision-making.

### 3.3.1. Sequential decision-making framework

We consider a general class of multiagent problems that can be modeled as partially observable stochastic games [20].

**Definition 8** (Partially observable stochastic game)**.** A partially observable stochastic game (POSG) is a tuple $(N, S, A, O, T, \Omega, R, h, b^0)$ where $N$ agents interact in the finite state space $S$. $A = A_1 \times \cdots \times A_N$ and $O = O_1 \times \cdots \times O_N$ are the finite spaces of joint actions and observations. $T$ models the transition probabilities as $T(s' \mid s, a) = \mathbb{P}(s' \mid s, a)$, namely the probability of resulting in state $s'$ when actions $a = (a_1, \ldots, a_N)$ are chosen in state $s$. $\Omega$ is the observation distribution, $\Omega(o' \mid a, s') = \mathbb{P}(o' \mid a, s')$, the probability of receiving observations $o = (o_1, \ldots, o_N)$ when performing actions $a$ results in state $s'$. $R(s, a) = (R_1(s, a), \ldots, R_N(s, a))$ specifies the immediate reward function for each agent for taking actions $a$ in state $s$. Finally, $h$ defines the process horizon and $b^0$ the initial state distribution.

In a POSG, each agent $i$ employs a policy $\pi_i$ that corresponds to a possibly stochastic map from the action-observation history to the action space. A joint policy $\pi = (\pi_1, \ldots, \pi_N)$ is a tuple of policies for each of the agents. Given a joint policy $\pi$, we can define the corresponding expected cumulative reward function for an agent $i$: $V_i^\pi = \mathbb{E}\left[\sum_{t=1}^h R_i^t \mid \pi, b^0\right]$.

Different optimization problems and corresponding solution concepts can be considered according to the adversarial or cooperative nature of the agents. In the first case, the problem consists of searching for a Nash equilibrium (NE)[21]: a joint policy $\pi = (\pi_i, \pi_{-i})$ such that each agent $i$ draws no advantage from deviating from its policy $\pi_i$ given the policies of the others $\pi_{-i}$. In cooperative settings modeled by decentralized partially observable Markov decision processes (Dec-POMDPs) [22, 23], the solution consists of the optimal join policy $\pi$ maximizing the value function shared by all players. Many solutions methods for searching NE (e.g. fictitious play [24, 25], double oracle [26], parallel Nash memory [27]) and for optimal policies for Dec-POMDPs (e.g. influence search [5]) use best-response computations as their inner loop. A best response policy $\pi_i^*$ for agent $i$ against fixed policies $\pi_{-i}$ for the other agents maximizes the value function $\max_{\pi_i} V_i^{(\pi_i, \pi_{-i})}$.

However, computing best responses implies reasoning over the entire state space and other agents beliefs [10]. To ease this task we want to exploit the structural properties of the environment to build local abstractions. Frequently, in fact, the state space can be thought as composed of different state variables, or *factors* [28]. In this case, the model is called *factored POSG* and every state $s \in S$ can be represented as a tuple of factors instantiations $s = (x_1, \ldots, x_M)$. This structure allows to decompose the system into (weakly) coupled subproblems, the *local models*, for single agents including only few factors. Thus, the idea is to define an equivalent problem to the best response in which the agent needs only to reason over the factors included in the local model.

### 3.3.2. Influence-based abstraction

Influence-based abstraction formalizes the concept of the local model for an agent in a factored POSG and provides the formal framework of the local abstraction for the best-response problem.

To simplify the discussion, we restrict to locally fully-observable POSGs [29]. We use capital letters to denote state variables and small letters for variable instantiations.

**Definition 9** (Fully observable local model)**.** Given a factored POSG, a fully observable local model for an agent is a subset of fully observable state variables, called *modeled factors*, including all state variables directly affecting the observations and reward. We denote the *local state*, that is, the collection of all the modeled factors, as $X$ and the complementary collection of *non-modeled factors* as $Y$.

The idea of the local model is to exclude those factors that are not necessary to compute the best response from the subset of variables over which the agent needs to reason and that can therefore be abstracted away. In a local model, we distinguish between modeled factors that are only affected by other factors and actions that are modeled, called *only-locally affected factors*, and *non-locally affected factors* that are modeled factors affected by at least one factor or action of the external part. We denote by $X_{\text{loc}}$ the collection of the only-locally affected factors.

**Definition 10.** We refer to the non-modeled factors or external actions that directly exert an influence on at least one of the modeled factor as the *influence sources* $Y_{\text{src}}$. We define the *influence destinations* $X_{\text{dest}}$ as the modeled factors that directly experience the influence of the external part of the system through the influence sources.

Thus, the local state $X$ can be thought of as $X = (X_{\text{loc}}, X_{\text{dest}})$. In other words, the local model comprises only a few modeled-factors. Some of them are influenced by the external part and therefore are called influence destinations. The non-modeled factors or actions that directly exert this influence on the modeled factors are accordingly the sources of the influence. See Section 3.2 in [9] for the formal definitions of the general, not fully-observable, case.

When abstracting away the non-modeled factors $Y$, the dynamics of $X_{\text{dest}}$ become non-Markovian. Accordingly, the local state transitions are not well defined. To define the local state dynamics, we need to include as part of the abstract state space the history of relevant modeled actions and factors to infer the influence sources $Y_{\text{src}}$. The notion of *d-separating set* (d-set) [30] formalizes this concept. The d-set at time $t$, $D^t$ contains the histories of the modeled factors and actions necessary to predict the influence sources $Y^t_{\text{src}}$.

**Definition 11** (Influence point)**.** The exact influence point (EIP) $I = (I^0, \dots, I^{h-1})$ is a collection of conditional probability distributions $I^t(Y^t_{src} \mid D^t) \triangleq \mathbb{P}(Y^t_{src} \mid D^t, \pi_{-i}, b^0)$ of the influence sources $Y^t_{src}$ given the possible instantiations of the d-set.

We refer to Section 4.1 in [9] for extensive definitions of influence point and d-separating set. The local model dynamics is formalized by the *influence-augmented local model*, a factored Markov decision process (MDP) where the state space consists of the local state augmented with the d-set.

**Definition 12** (Influence augmented local model)**.** Consider a fully observable local model for agent $i$ in a factored POSG $(n, S, A, O, T, \Omega, R, h, b^0)$, that identifies modeled factors $X$. Fix the policies $\pi_{-i}$ for the other agents. An influence augmented local model (IALM) $\mathcal{M}_i = (\bar{S}_i, A_i, T_i, R_i, h, b^0)$ is a factored MDP where the action space $A_i$ and rewards $R_i$ correspond to the POSG agent $i$'s actions and rewards. The augmented state space $\bar{S}_i$ consists of local states and d-separating sets as $\bar{s}^t = (x^t, d^t) = ((x^t_{\text{loc}}, x^t_{\text{dest}}), d^t)$. The transition functions are derived as

$$T_i(\bar{s}^{t+1} \mid \bar{s}^t, a^t) = T(x^{t+1}_{\text{loc}} \mid x^t, a^t) \sum_{y^t_{src}} T(x^{t+1}_{\text{dest}} \mid x^t, y^t_{src}, a^t)\, I^t(y^t_{src} \mid d^t) \tag{3.1}$$

This model defines an equivalent problem to the best-response problem for agent $i$ against $\pi_{-i}$. Namely, the optimal policy for the MDP defined by the IALM $\mathcal{M}_i$ corresponds to the best response against policies $\pi_{-i}$ in the factored POSG. The claim and complete proof that IBA provides a lossless abstraction can be found in Section 6 [9].

### 3.3.3. INFLUENCE IN PLANETARY EXPLORATION

To give a concrete intuition of influence-based abstraction, we use a version of the planetary exploration environment [5]. A rover has to explore a planet and its navigation may be guided by a plan from a satellite. The goal of the rover is to move until it reaches a target site, where it will collect a positive reward. However, any time it fails to step forward, it will receive a penalty. The satellite might help the rover by providing a plan which increases the likelihood of a successful step.

Figure 3.1 shows a dynamic Bayesian network (DBN) [28] that compactly represents the problem. At each time $t$ the rover can observe its position $pos$ and if a plan $pl$ was available at the previous time step. The rover's local state consists of $X = (pos, pl)$. The plan $pl$ is the influence destination since it is directly affected by the non-modeled satellite action $a_{\text{sat}}$ which is accordingly the influence source. Contrarily, the position $pos$ is only directly affected by local variables. The decisions of the satellite might depend on its level of battery, the *charge*, which only indirectly affects the local model through the satellite actions. Potentially, the decision-making problem of the satellite might depend on a conceivably larger number of variables: it has to manage its own resources, send plans to other rovers etc.

However, the only information the rover needs to retrieve to act optimally is whether the satellite will make available a routing plan at the next step. Therefore, it can abstract away all the other state variables and try to *infer* the satellite's action, given all the relevant information that it has in the local model. In this scenario, it turns out that all it has to remember is the history of the availability of plans at each time step. Therefore, the influence point consists of the distribution of the satellite's actions $Y^t_{src} = a^t_{\text{sat}}$, given the local history of the plan $D^t = (pl^0, \dots, pl^t)$.

## 3.4. Planning with approximate influence representations

IBA has the potential to enable faster planning in complex domains where using the entire model would typically be too heavy and computational demanding with no information loss. However, exact influence computation requires solving a large number of intractable inference problems. In fact, in most applications, the size of the d-set increases linearly in time, resulting in an exponentially increasing number of instantiations of the d-set. This means that representing an EIP takes exponential space, and computing an EIP requires solving exponentially many and possibly hard inference problems. This motivates the idea of using approximate influence point (AIP) representations: we instead use advances in machine learning that allow us to generalize AIPs over d-separating sets. The overall idea consists of transforming the global model into a new approximate influence-augmented local model at which any solution method can be applied. Our method is therefore complementary to the planning method. The advantage lies in the reduction of the problem size and therefore in computational complexity.

The approach is straightforward but sketched in Algorithm 1 for completeness: first we collect samples of the d-set $D^t$, the local history sufficient to predict the influence sources $Y^t_{\text{src}}$ in the global model using an exploratory policy $\pi_i^{\text{Exp}}$ for agent $i$ (lines 1-3). In the second step, we learn the approximate influence point $\hat{I}^t$ using the $n$ trajectories of $\left(D^t, Y^t_{\text{src}}\right)$ as training set (lines 4-5). Then, we construct the approximate local transitions $\hat{T}_i$ using the learned influence point $\hat{I}$ according to

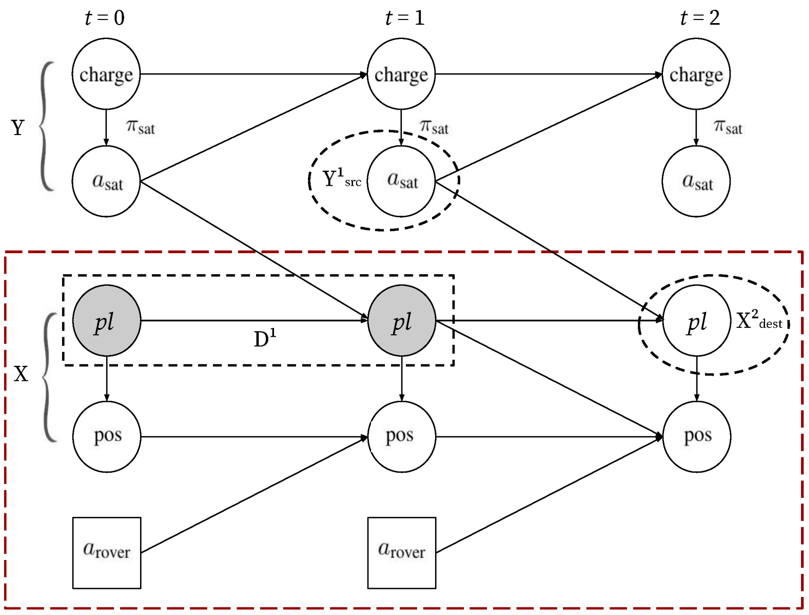


Figure 3.1.: A DBN of the planetary exploration domain. The dotted red square delimits the local model of the rover, including the modeled factors $X$. The external part comprises the non-modeled factors $Y$. Among them, the satellite action $a_{\text{sat}}$ directly affects the availability of a plan $pl$ and therefore is the influence source. The gray circles constitute the d-set $D^1$ to infer the action $a_{\text{sat}}$ at time $t = 1$. Namely, the history of the plan $(pl^0, pl^1)$ retains the sufficient local information to infer the satellite action $a^1_{\text{sat}}$. The influence on the local model at the next time step $t = 2$ is experienced directly by the modeled factor plan $pl^2$ that is thus the influence destination.

3.1 (line 6). Then, we assume that the resulting local problem might be solved by simulation-based planning or RL approaches to derive a near optimal best-response $\hat{\pi}^*_i$ for agent $i$ (line 7-9). Note that in line 8 any solution method can be used to solve the MDP problem. The idea is that this phase uses $m \gg n$ episodes, thus making the initial cost of using the expensive global simulations negligible.

While planning with a learned approximate influence could lead to significant speedups, there is also a risk as the approximate influence point may induce inaccurate state transitions, leading to a loss in value. A key question therefore is: what constitutes good AIPs? What conditions need to hold on $\hat{I}$, for $\hat{\pi}^*_i$ to be close to optimal?

**Algorithm 1** Exact Best Response to Approximate Influence

```
Require: exploratory policy $\pi_i^{\mathrm{Exp}}$, policies $\pi_{-i}$
1: {Simulation:}
2: Create a dataset of $n$ trajectories of the global model
3: Extract $\left(d_k^t, y_{\mathrm{src},k}^t\right)_{k=1:n,t=1:h}$.
4: {Train a Prediction Algorithm:}
5: Learn approximate influence $\hat{I}^t$ from $\left(d_k^t, y_{\mathrm{src},k}^t\right)_{k=1:n}$ for any $t = 0,\dots,h$
6: Compute approximate local transitions $\hat{T}_i$ from $\hat{I}$
7: {Solve the local planning/RL problem:}
8: Compute $\hat{\pi}_i^* = \mathrm{MDPSolver}(\hat{T}_i)$
9: return $\hat{\pi}_i^*$
```

## 3.5. Theoretical loss bounds

Here we want to discuss guarantees on the value loss when using AIPs instead of EIPs to derive best-response policies. Such kind of results allow us to derive conditions for approximate influence to yield near-optimal solutions.

Formally, we consider two IALMs $\mathcal{M} = (\bar{S}, A, T, R, h, b^0)$ and $\hat{\mathcal{M}} = (\bar{S}, A, \hat{T}, R, h, b^0)$ sharing the same augmented state space, action space, and rewards. They differ only in the transition functions $T$ and $\hat{T}$ that are induced, respectively, by the EIP $I$ and an AIP $\hat{I}$ according to 3.1. We see $\hat{\mathcal{M}}$ as an approximation to $\mathcal{M}$. We omit the subscript $i$ when referring to an IALM for agent $i$ to simplify the notation.

### 3.5.1. $L^1$ loss bound

We introduce first the necessary notation. We use $V^*$ to denote the objective optimal value for $\mathcal{M}$. $\hat{\pi}^*$ refers to the optimal policy in the approximate IALM $\hat{\mathcal{M}}$. $V^{\hat{\pi}^*}$ denotes the value achieved by policy $\hat{\pi}^*$ in the IALM $\mathcal{M}$. Our aim is to bound the value loss given by the difference $V^* - V^{\hat{\pi}^*}$. We define $|R| \triangleq \max_{a,s}|R(s,a)|$. We use $\|.\|_1$ and $D_{KL}$ to denote the $L^1$-norm and the KL divergence between probability distributions.

The proof of the value loss bound derived is divided into two steps. We proceed first by bounding the value loss with the difference between the IALM transitions functions $\|T - \hat{T}\|_1$. Then, we prove an upper bound for $\|T - \hat{T}\|_1$ in terms of the difference $\|I - \hat{I}\|_1$. Finally, combining these two results we derive an upper bound for the value loss in terms of the distance between the EIP and the AIP $\|I - \hat{I}\|_1$. We collect the proofs of all the following claims in Appendix B.

Since the IALM $\hat{\mathcal{M}}$ is essentially an approximation of the exact influence IALM $\mathcal{M}$ with imprecise transitions, we apply the loss bounds for MDPs with uncertain transition probabilities [31–33] to the case of the IALMs.

**Theorem 2.** Consider the two IALMs $\mathcal{M}$ and $\hat{\mathcal{M}}$, the following loss bound holds

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \le 2h^2|R| \max_t \max_{\bar{s}^t, a^t} \|(T(\cdot \mid \bar{s}^t, a^t) - \hat{T}(\cdot \mid \bar{s}^t, a^t)\|_1 \tag{3.2}$$

The importance is that we can bound the value loss when the IALM transition functions $T$, $\hat{T}$ induced by $I$, $\hat{I}$ are sufficiently close.

For the second step, we use the relation between the IALM transitions $T$ and the influence point $I$ expressed by 3.1.

**Lemma 1.** Consider the IALMs transitions $T$ and $\hat{T}$. For any $t$, augmented state $\bar{s}^t = (x^t, d^t)$ and action $a^t$

$$\|(T(\cdot \mid \bar{s}^t, a^t) - \hat{T}(\cdot \mid \bar{s}^t, a^t)\|_1 \leq \|(I^t(\cdot \mid d^t) - \hat{I}^t(\cdot \mid d^t)\|_1 \tag{3.3}$$

Combining Theorem 2 and Lemma 1, we obtain the $L^1$ loss bound.

**Theorem 3.** Consider an IALM $\mathscr{M} = (S, A, T, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathscr{M}} = (S, A, \hat{T}, R, h, b^0)$. Then, a value loss bound in terms of the $L^1$-norm error is given by

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \leq 2h^2|R| \max_{t,d^t} \|(I^t(\cdot|d^t) - \hat{I}^t(\cdot \mid d^t)\|_1 \tag{3.4}$$

This shows that the value loss is bounded by the worst case $L^1$-distance between the exact and the approximate influence over all the possible d-set instantiations.

### 3.5.2. KL DIVERGENCE BOUND

We now present a loss bound in terms of the KL divergence between the approximate and the exact influence. Since the cross entropy and KL divergence differ solely by an additive constant, this result establishes a relation between the value loss and the cross entropy error for influence approximation.

**Corollary 1.** Consider an IALM $\mathscr{M} = (S, A, T, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathscr{M}} = (S, A, \hat{T}, R, h, b^0)$. Then, a value loss bound in terms of KL divergence error is given by

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \leq 2h^2|R| \max_{t,d^t} \sqrt{2D_{KL}(I^t(\cdot \mid d^t) \| \hat{I}^t(\cdot \mid d^t))} \tag{3.5}$$

We see that the performance loss can be bounded by the max KL divergence. Cross entropy loss optimizes the mean of such loss $\mathbb{E}_{t,D^t}\left[D_{KL}(I^t(\cdot \mid D^t)) \| \hat{I}^t(\cdot \mid D^t)\right]$ and therefore is aligned with this bound: even though it does not optimize the max itself, it is intuitively clear that in many problems a low mean will imply a low max error, even the more since the mean is known to be sensitive to outliers. Moreover, this is an asymptotically tight bound. That is, when the KL divergence tends to zero the approximate influence solution approaches the true optimal one. This suggests that neural networks learning approaches are well suited for the approximate influence learning task and that the cross entropy, which has been widely used as test error, can give a priori insight on the value loss.

### 3.5.3. PROBABILISTIC LOSS BOUND

The previous section gives an idea of what AIPs properties can guarantee bounded value loss. However, the direct application of Theorem 3 and Corollary 1 requires the distance to the true influence point $I$, which is unknown. In this section, we present an approach to overcome this limitation: a probabilistic loss bound depending only on the distance to an empirical influence distribution, which can be measured using a test set.

Precisely, assume that for a given instatiation $d^t$ of the d-set at time $t$ we have $n$ samples of the influence sources $\left\{y^t_{\text{src},1},\dots,y^t_{\text{src},n}\right\}$ and let

$$I^t_n(y^t_{\text{src}} \mid d^t) = \frac{1}{n}\sum_{k=1}^{N}\delta_{y^t_{\text{src},k}}(y^t_{\text{src}})$$

denote the empirical conditional distribution. Our result presents a value loss bound in terms of the distance between the empirical influence $I^t_n$ and the approximate $\hat{I}^t$.

**Theorem 4.** Consider an IALM $\mathcal{M} = (S, A, T, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathcal{M}} = (S, A, \hat{T}, R, h, b^0)$. Assume that for every time $t$ and d-set instantiation $d^t$, we have at least an sample of size $n$ of influence sources. Then for every $\varepsilon > 0$

$$\mathbb{P}\left(\|V^* - V^{\hat{\pi}^*}\|_\infty \le 2h^2|R|\max_{t,d^t}\|(I^t_n(\cdot \mid d^t) - \hat{I}^t(\cdot \mid d^t)\|_1 + \varepsilon\right) \ge 1 - h|D^h|(2^{|Y_{\text{src}}|} - 2)e^{-n\frac{\varepsilon^2}{2}} \tag{3.6}$$

where $|Y_{\text{src}}|$ is the cardinality of the influence sources space and $|D^h|$ the cardinality of the d-sets space at time $h$.

This theorem states that, with high probability, the value loss of using an AIP $\hat{I}$ is upper bounded by the empirical error (that we can measure using a test set). We believe that this is a first step towards bounds that can be used in practical applications.

However, the bound in Theorem 4, still requires an intractable maximization over all possible instantiations of d-sets (and needs $n > \log(h|D^h|2^{|Y_{\text{src}}|})$ samples for each of these). Moreover, the maximization over d-sets might be too conservative in many cases. Namely, the right-hand side of (3.5) corresponds to the loss one agent would incur if every time step the mistake is caused by the worst-case approximation of the influence over d-sets. We think that the maximization may be replaced with an appropriate expectation over instantiations. In this respect, identifying versions of these bounds that work with expectations over d-sets, rather than maximization, is one of the important directions of future work that our paper identifies.

## 3.6. Multiagent implications

So far, we have focused on the perspective of a single agent, that computes a best response against the fixed policies of other agents. However, the results in this paper can have important implications also for settings where we optimize or solve for the policies of multiple agents at the same time. In fact, the insights of influence-based abstraction originated from the study of multiagent systems [5–8, 34–40], so these implications should hardly be surprising. Nevertheless, we think it is useful to spell out some of these implications and will do so in the remainder of this section.

First, we point out that the best-response setting that we consider is very general. In fact, the notion of a 'fixed policy' of an agent in a POSG is very powerful: such a fixed policy is a mapping from histories of actions and observations to distributions over actions and therefore powerful enough to model learning agents. This means that the fixed policies that we compute a best response against could include, e.g., a Q-learning agent.

A second implication is for the computation of equilibria in POSGs. Many methods for computing Nash equilibria in POSGs [24–27, 41], use best-response computation as an inner loop. In many cases, such as in the double-oracle algorithm [26] or the parallel Nash Memory [27], replacing such a best-response computation with an $\epsilon$-best response computation can enable us to compute $\epsilon$-approximate Nash Equilibria ($\epsilon$-NEs) [21], in which no player can benefit more than $\epsilon$ from deviating. As such, our results–that show under what conditions on the influence predictions we can compute an $\epsilon$-approximate best response–may lead to computationally feasible paths to compute $\epsilon$-NEs.

Even in cases where computing $\epsilon$-NEs will remain out of reach, our results can provide insight in complex MAS.

**Observation 1.** Given a method $E$ to estimate an AIP with an error (maximum $L_1$ norm) of at most $\epsilon_1$, we can verify if a particular joint policy $\pi = (\pi_1, \dots, \pi_N)$ is an $(2h^2|R|\epsilon_1)$-NE as follows:

1. use $E$ to estimate the influence $\hat{I}_i$ on each agent $i$
2. verifying if $\pi_i$ is an optimal solution in the IALM constructed for agent $i$.

Of course, the question of how to develop such estimators $E$ is not trivial in the general case, but there are many special cases with compact influence descriptions where this is possible [9, 42].

Moreover, IBA serves as the basis for influence search [5, 7, 34] in cooperative settings. The key idea is that an influence point captures all the relevant information about many different policies of the other agents. That is, different joint policies might induce the same influences on the local models of the agents. Thus, the space of 'joint influence points' $I = (I_1, \dots, I_N)$ can be much smaller than the space of joint policies $\pi = (\pi_1, \dots, \pi_N)$, and it can be much more efficient to search through the former. [40].[1]

## 3.7. Empirical evaluation

The bounds presented in Section 3.5 establish a relation between the value loss and the cross entropy of the influence approximations. In particular, they suggest that the *mean* cross entropy loss is aligned with the objective of minimizing the performance loss.

[1]Roughly speaking, an influence search algorithm searches in the space of possible joint influences, and for each joint influence point $I = (I_1, \dots, I_n)$, it solves a local constrained best-response problem for each agent (possibly in parallel): each agent $i$ computes a best-response $\hat{\pi}^*(I)$ to $I_i$ constrained to inducing influences $I_{-i}$ on the local models of the other agents.

Here we evaluate if this alignment translates into practical settings by investigating the relation between the mean cross entropy

$$\mathrm{CE}(I,\hat{I}) \triangleq \mathbb{E}_{t,D^t}\left[\ \mathrm{CE}(I^t(\cdot \mid D^t), \hat{I}^t(\cdot \mid D^t))\right]$$

and the performance loss $V^* - V^{\hat{\pi}^*}$. Namely, we expect to see that a better influence approximation in terms of cross entropy loss, indeed, leads to a smaller value loss. Here we try to validate this hypothesis.

Similarly, we investigate to what extent the *mean* $L^1$-norm error

$$\|I-\hat{I}\|_1 \triangleq \mathbb{E}_{t,D^t}\left[\|I^t(\cdot \mid D^t) - \hat{I}^t(\cdot \mid D^t)\|_1\right]$$

allows to a priori assess the quality of the approximations in terms of the value loss. In fact, according to the results in Section 3.5, the $L^1$-norm has the potential to provide a tighter bound for the performance loss. If experimentally confirmed, this would motivate future work to investigate the possibility to use different training losses based on $L^1$ distance.

### 3.7.1. EXPERIMENTAL SETUP

We follow the procedure sketched in Algorithm 1. That is, we first run the simulations from the global model using an exploratory random policy $\pi^{\mathrm{Exp}}$ for the local agent and the set of fixed policies for the other agents. We collect the samples of influence sources and d-sets $\{y^t_{\mathrm{src,k}}, d^t_k\}_{t=1:h,k=1:n}$. We train a LSTM neural network with cross entropy loss to induce the approximate influence $\hat{I}^t(\cdot \mid D^t)$. At different training epochs, for the corresponding approximations of the influence, we build the approximate-influence local model. Then, we compute the optimal policy $\hat{\pi}^*$ in the approximate-influence model through value iteration [43]. We evaluate the policy $\hat{\pi}^*$ in the global model using $M$ simulations to obtain the value achieved $V^{\hat{\pi}^*}$. Note that since we use an exact solution method to compute the policy $\hat{\pi}^*$ in the approximate-influence model, any other solution method would get lower performance in the local model. For each epoch, to assess the quality of the approximation, we use Monte Carlo estimators for the mean cross entropy $\mathrm{CE}(I,\hat{I})$ and the $L^1$-norm error $\|I-\hat{I}\|_1$, computed using a test set as

$$\begin{aligned} error_{\mathrm{CE}} &= -\frac{1}{h}\sum_{t=1}^{h}\frac{1}{n}\sum_{k=1}^{n}\ln\left(\hat{I}^t(y^t_{\mathrm{src,k}} \mid d^t_k)\right) \\ error_{\mathrm{norm1}} &= \frac{1}{h}\sum_{t=1}^{h}\frac{1}{n}\sum_{k=1}^{n}\left\|\delta_{y^t_{\mathrm{src,k}}}(\cdot) - \hat{I}^t(\cdot \mid d^t_k)\right\|_1 \end{aligned} \tag{3.7}$$

We compute the average and standard deviation of the $error_{\mathrm{CE}}$ and $error_{\mathrm{norm1}}$ over different iterations of the same experiment. For any iteration, we repeat entirely the steps described above: we recollect training and testing samples from the global model, retrain the neural network etc. When the problem is sufficiently easy to solve, we compute the optimal value $V^*$ and then we measure the correlation between $error_{\mathrm{CE}}$, $error_{\mathrm{norm1}}$ and the value loss $V^* - V^{\hat{\pi}^*}$. Otherwise, we measure the correlation between the test errors and $-V^{\hat{\pi}^*}$. In fact, since the optimal value $V^*$ is constant for increasing epochs, it does not affect the correlation. That is, $Corr(error, V^* - V^{\hat{\pi}^*}) = Corr(error, -V^{\hat{\pi}^*})$.

We run the experiments in three domains: a version of the planetary exploration [5], a traffic domain and the fire fighters problem [44].

**Traffic network.** In this domain, we simulate a traffic network with 4 intersections (see Figure 3.2). The sensors of the traffic lights at each intersection provide information on the $3 \times 3$ local grid around them. In Figure 3.2, the dotted red square represents the local model for the protagonist agent. The other traffic lights employ hand-coded policies prioritizing fixed lanes. The goal of the agent is to minimize the total number of vehicles waiting at the local intersection. In order to act optimally, the local agent only needs to predict if there will be incoming cars from the *east* and *north* lanes of the local model the next time step. The outgoing cars have some probability to re-enter in the network from other lanes. That is, the outgoing vehicles from *south* and *west* can affect the decisions of other traffic lights and consequently, the vehicles inflow in the local model at future time steps.

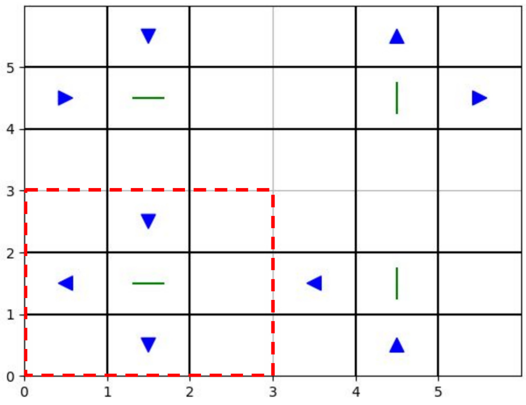
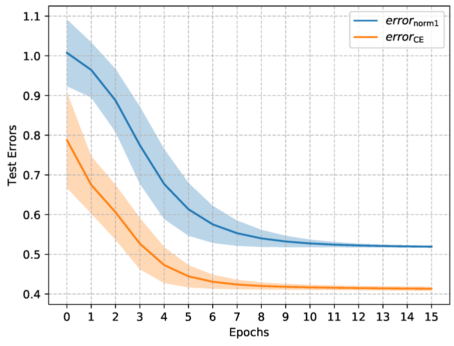


Figure 3.2.: Traffic network. The local model is delimited by the dotted square. The blue triangles represent the vehicles and the green bars the traffic lights.

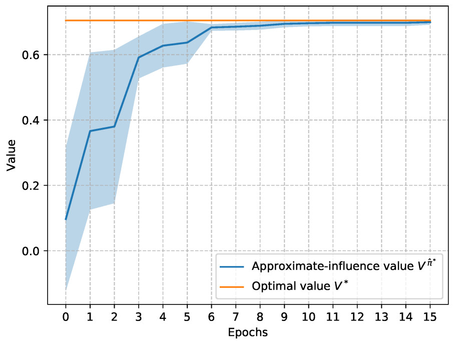
**Fire fighters.** We model a team of 2 agents that need to cooperate to extinguish fires in a row of 3 houses. At every time step, every agent can choose to fight fires at one of its 2 neighboring houses. The goal of each agent is to minimize the number of neighboring houses that are burning. We take the perspective of one agent with a local model including only the two neighboring houses.

## 3.7.2. EXPERIMENTAL RESULTS

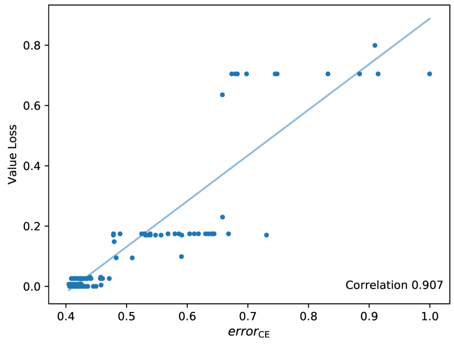
Figures 3.3(a), 3.4(a), 3.5(a) show the test errors measured by $error_{\mathrm{CE}}$ and $error_{\mathrm{norm1}}$ (3.7), as functions of the training epochs for the three domains. They show



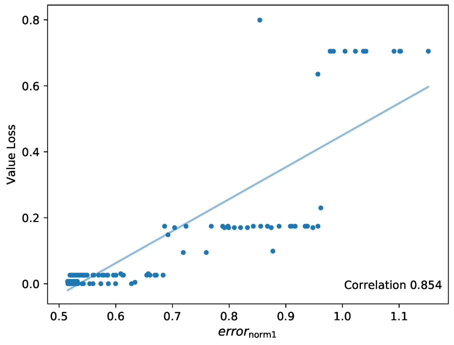
(a) Test errors for increasing training epochs.



(b) Value achieved by $\hat{\pi}^*$ compared to the optimal value for increasing training epochs.



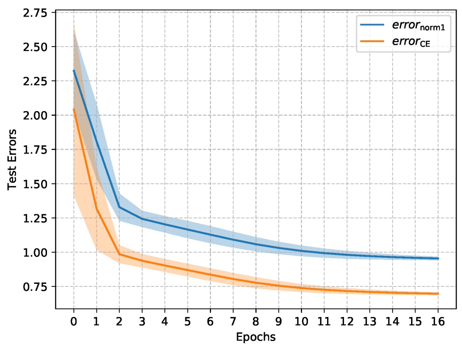
(c) Correlation between value loss and $error_{\mathrm{CE}}$.



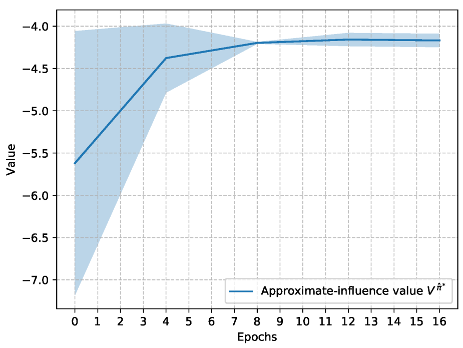
(d) Correlation between value loss and $error_{\mathrm{norm1}}$.

Figure 3.3.: Planetary exploration.

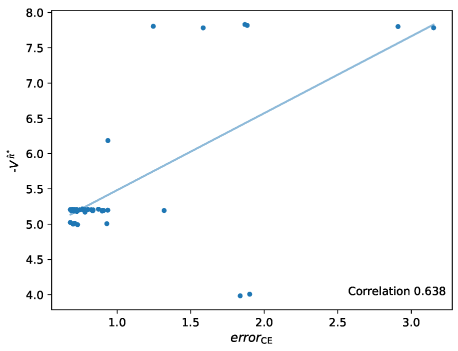
that performance in terms of cross entropy and $L^1$-norm test errors improves monotonically with the number of epochs. These trends suggest that the AIP improves with training, the question is if this also corresponds to a value loss decrease. Looking at Figure 3.3(b) this seems to be the case in the planetary exploration setting. We see that the performance of the policy derived from the approximate-influence IALM is very close to optimal from 6 epochs onward, right about when the training of the neural network starts to stagnate. Moreover, Figure 3.3(c) and 3.3(d) show that the decrease in mean empirical cross entropy or $L^1$-norm error indeed correlates well with actual value loss. In the traffic domain, we also see that there is a significant improvement in the value achieved by the approximate-influence policy $\hat{\pi}^*$ for increasing number of epochs in Figure 3.4(b). The test errors seem still well aligned with the value loss in Figures 3.4(c), 3.4(d). For



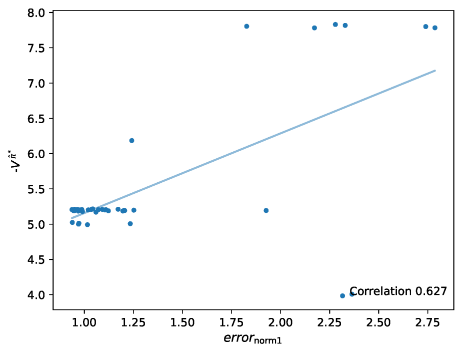
(a) Test errors for increasing training epochs.



(b) Value achieved by $\hat{\pi}^*$ for increasing training epochs computed every 4 epochs.



(c) Correlation between value loss and $error_{\text{CE}}$.



(d) Correlation between value loss and $error_{\text{norm1}}$.

Figure 3.4.: Traffic network.

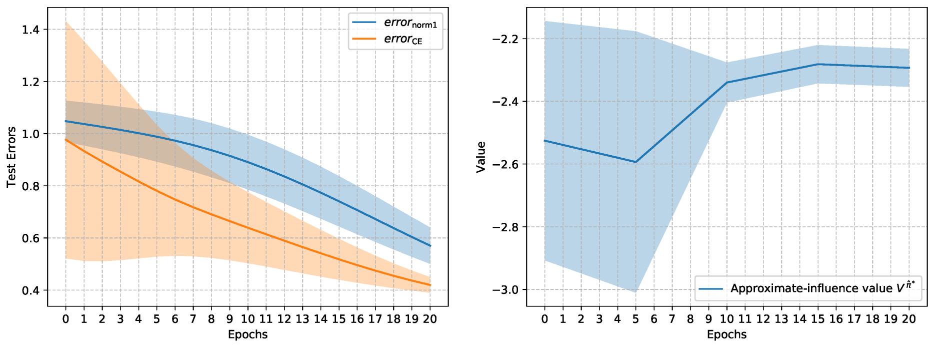
the fire fighters domain, Figure 3.5(b) shows that the value improves over training epochs coherently with the decrease of the test errors.

For more results on the correlation analysis for different planetary exploration settings, see Appendix B.

We can conclude that both the empirical cross entropy error and the $L^1$-norm errors correlate well with the value loss and therefore provide a priori insight on the quality of the influence approximation in terms of the value achieved.

## 3.8. Conclusions and discussion

In this paper, we provided a priori quality guarantees for the value loss resulting from approximating the influence point. Our results show that the objective of a



(a) Test errors for increasing training epochs.

(b) Value achieved by $\hat{\pi}^*$ for increasing training epochs computed every 5 epochs.

Figure 3.5.: Fire fighters problem.

neural network trained with the cross entropy loss is well aligned with the goal of minimizing the value loss. Our empirical results demonstrate that empirical (mean) estimators of the cross entropy and the $L^1$-norm are predictive of the value loss. We used this insight to evaluate the transfer of our theoretical results to practical scenarios.

In this work, we limit the discussion to fully observable local models. However, in the general case, we would be dealing with a local POMDP. We believe that a generalization of our bound in the partially observable case can be obtained for instance leveraging the 'back-projected value vectors' formulation [35]. As future work, we intend to further investigate this.

The current formulation of the loss bounds involves a maximization over the space of all the possible instantiations of the d-set, which might grow exponentially in time. In future work, we would like to derive expressions that do not need a maximization, but for example are in expectation over a sampling process. This would lead to tighter bounds, which could be used in practice for model selection or value loss estimation.

# REFERENCES

[1] E. Congeduti, A. Mey, and F. Oliehoek. “Loss bounds for approximate influence-based abstraction”. In: *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2021.

[2] R. Givan, S. Leach, and T. Dean. “Bounded-parameter Markov decision processes”. In: *Artificial Intelligence* (2000), pp. 71–109.

[3] L. Li, T. J. Walsh, and M. L. Littman. “Towards a unified theory of state abstraction for MDPs”. In: *International Symposium on Artificial Intelligence and Mathematics (AI&Math)*. 2006.

[4] A. Bai, S. Srivastava, and S. Russell. “Markovian state and action abstractions for MDPs via hierarchical MCTS.” In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2016.

[5] S. Witwicki and E. Durfee. “Influence-based policy abstraction for weakly-coupled Dec-POMDPs”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2010.

[6] R. Becker, S. Zilberstein, and V. Lesser. “Decentralized Markov decision processes with event-driven interactions”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2004.

[7] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. “Solving transition independent decentralized Markov decision processes”. In: *Journal of Artificial Intelligence Research (JAIR)* (2004), pp. 423–455.

[8] P. Varakantham, J.-y. Kwak, M. E. Taylor, J. Marecki, P. Scerri, and M. Tambe. “Exploiting coordination locales in distributed POMDPs via social model shaping”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2009.

[9] F. Oliehoek, S. Witwicki, and L. Kaelbling. “A sufficient statistic for influence in structured multiagent environments”. In: *Journal of Artificial Intelligence Research (JAIR)* (2021), pp. 789–870.

[10] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. “Taming decentralized POMDPs: towards efficient policy computation for multiagent settings”. In: *International Joint Conference on Artificial Intelligence, (IJCAI)*. 2003.

[11] F. Oliehoek and C. Amato. “Best response Bayesian reinforcement learning for multiagent systems with state uncertainty”. In: *AAMAS Workshop on Multiagent Sequential Decision Making Under Uncertainty*. 2014.

[12] A. Panella and P. Gmytrasiewicz. “Interactive POMDPs with finite-state models of other agents”. In: *Autonomous Agents and Multi-Agent Systems (JAAMAS)* (2017), pp. 861–904.

[13] F. Karim, S. Majumdar, H. Darabi, and S. Chen. “LSTM fully convolutional networks for time series classification”. In: *IEEE Access* (2017), pp. 1662–1669.

[14] C. Lea, M. Flynn, R. Vidal, A. Reiter, and G. Hager. “Temporal convolutional networks for action segmentation and detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[15] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. “An actor-critic algorithm for sequence prediction”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[16] R. Dearden and C. Boutilier. “Abstraction and approximate decision-theoretic planning”. In: *Artificial Intelligence* (1997), pp. 219–283.

[17] M. Petrik and D. Subramanian. “RAAM: the benefits of robustness in approximating aggregated MDPs in reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[18] D. Abel, D. Hershkowitz, and M. Littman. “Near optimal behavior via approximate state abstraction”. In: *International Conference on Machine Learning (ICML)*. 2016.

[19] T. Dean, R. Givan, and S. Leach. “Model reduction techniques for computing approximately optimal solutions for Markov decision processes”. In: *Uncertainty in Artificial Intelligence (UAI)*. 1997.

[20] E. Hansen, D. Bernstein, and S. Zilberstein. “Dynamic programming for partially observable stochastic games”. In: *AAAI Conference on Artificial Intelligence*. 2004.

[21] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[22] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. “The complexity of decentralized control of Markov decision processes”. In: *Mathematics of Operations Research* (2002), pp. 819–840.

[23] F. Oliehoek and C. Amato. *A concise introduction to decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer, 2016.

[24] U. Berger. “Brown’s original fictitious play”. In: *Journal of Economic Theory* (2007), pp. 572–578.

[25] G. Brown. “Iterative solution of games by fictitious play”. In: *Activity analysis of production and allocation* (1951), pp. 374–376.

[26] B. McMahan, G. Gordon, and A. Blum. “Planning in the presence of cost functions controlled by an adversary”. In: *International Conference on Machine Learning (ICML)*. 2003.

[27] F. Oliehoek, E. De Jong, and N. Vlassis. “The parallel Nash memory for asymmetric games”. In: *Conference on Genetic and evolutionary computation*. 2006.

[28] C. Boutilier, T. Dean, and S. Hanks. “Decision-theoretic planning: structural assumptions and computational leverage”. In: *Journal of Artificial Intelligence Research (JAIR)* (1999), pp. 1–94.

[29] C. Goldman and S. Zilberstein. “Decentralized control of cooperative systems: categorization and complexity analysis.” In: *Journal of Artificial Intelligence Research (JAIR)* (2004), pp. 143–174.

[30] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[31] K. Delgado, S. Sanner, and L. De Barros. “Efficient solutions to factored MDPs with imprecise transition probabilities”. In: *Artificial Intelligence* (2011), pp. 1498–1527.

[32] A. Strehl, L. Li, and M. Littman. “Reinforcement learning in finite MDPs: PAC analysis.” In: *Journal of Machine Learning Research (JMLR)* (2009).

[33] A. Mastin and P. Jaillet. “Loss bounds for uncertain transition probabilities in Markov decision processes”. In: *IEEE Conference on Decision and Control*. IEEE. 2012.

[34] S. Witwicki, F. Oliehoek, and L. Kaelbling. “Heuristic search of multiagent influence space”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2012.

[35] F. Oliehoek, M. Spaan, and S. Witwicki. “Factored upper bounds for multiagent planning problems under uncertainty with non-factored value functions”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2015.

[36] M. Petrik and S. Zilberstein. “A Bilinear Programming Approach for Multiagent Planning”. In: *Journal of Artificial Intelligence Research (JAIR)* (2009), pp. 235–274.

[37] S. Witwicki and E. H. Durfee. “Flexible approximation of structured interactions in decentralized Markov decision processes”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2009.

[38] S. Witwicki and E. Durfee. “From policies to influences: a framework for nonlocal abstraction in transition-dependent Dec-POMDP agents”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2010.

[39] P. Velagapudi, P. Varakantham, P. Scerri, and K. Sycara. “Distributed model shaping for scaling to decentralized POMDPs with hundreds of agents”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2011.

[40] S. Witwicki and E. Durfee. “Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs”. In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2011.

[41] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, and T. Graepel. “A unified game-theoretic approach to multiagent reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[42] R. Chitnis and T. Lozano-Pérez. “Learning compact models for planning with exogenous processes”. In: *Conference on Robot Learning*. 2020.

[43] D. Bertsekas. *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995.

[44] F. Oliehoek. *Value-based planning for teams of agents in stochastic partially observable environments*. Amsterdam University Press, 2010.

# 4

# INFLUENCE LEARNING IN COMPLEX SYSTEMS

*High sample complexity hampers the successful application of reinforcement learning methods, especially in real-world problems where simulating complex dynamics is computationally demanding.* Influence-based abstraction (IBA) *was proposed to mitigate this issue by breaking down the global model of large-scale distributed systems, such as traffic control problems, into small local sub-models. Each local model includes only a few state variables and a representation of the* influence *exerted by the external portion of the system. This approach allows converting a complex simulator into local lightweight simulators, enabling more effective applications of planning and reinforcement learning methods. However, the effectiveness of IBA critically depends on the ability to accurately approximate the influence of each local model. While there are a few examples showing promising results in benchmark problems, the question of whether this approach is feasible in more practical scenarios remains open. In this work, we take steps towards addressing this question by conducting an extensive empirical study of learning models for influence approximations in various realistic domains, and evaluating how these models generalize over long horizons. We find that learning the influence is often a manageable learning task, even for complex and large systems. Additionally, we demonstrate the efficacy of the approximation models for long-horizon problems. By using short trajectories, we can learn accurate influence approximations for much longer horizons.*

## 4.1. INTRODUCTION

Controlling large distributed systems is a key challenge in artificial intelligence with the potential to impact many fields of application, including computing and information technology [2], energy systems [3, 4] and transportation [5]. Reinforcement learning (RL) could be promising for such problems as it provides a

This chapter has been published on Transaction on Machine Learning Research (TMLR), (2025) [1].

framework for studying how agents learn and plan under uncertainty in sequential decision-making problems. Despite recent successes for large sequential decision-making problems [6–9], RL techniques still suffer from high sample complexity [10, 11], which means that agents typically require many trajectories sampled from a simulator to achieve optimal performance. This sample inefficiency becomes a significant hurdle in real-world scenarios characterized by large, structured environments with complex dynamics. In these contexts, running computationally expensive simulators to collect a sufficient sample of trajectories can be unfeasible.

To mitigate this, influence-based abstraction (IBA) [12] offers a principled framework for decomposing the global model of a large factored multiagent problem into small local models, which support lightweight simulations. This approach has proven to be a powerful tool for accelerating online planning [13, 14], deep RL [15, 16], and deep multiagent RL [17]. The core idea is to leverage the factored structure of the environment to build a local model for each single agent, assuming that the other agents' policies are fixed. In this way, each agent has a local best response model, which it can use to compute a best response, in terms of local rewards, to the other agents[1]. By abstracting away a large portion of the environment, only a few state variable, referred to as the *local factors*, are retained in each local model. To overcome information loss due to the abstraction process, each local model is complemented with a representation of the *influence*, capturing the effects of external factors and policies of other agents on the dynamics of the local factors.

This abstraction approach has the potential to substantially reduce the state space of the best response problem without any loss in value. In other words, the local agent achieves the same performance when looking for solutions in the smaller local model as in the global model [19]. However, when part of the system is abstracted away, the local factors no longer preserve the Markov property. To restore Markov transitions, the state of the local model needs to be augmented with the history of appropriate state and action variables. Unfortunately as the number of local histories grows exponentially with time, computing the influence (a conditional marginal inference problem) becomes unfeasible, even for small systems or short-horizon problems. This motivates the idea of employing machine learning methods, which have shown excellent results in sequence modeling [20–22], to learn approximate representations of the influence. Intuitively, the more accurate the influence learning models are, the smaller the value loss. Congeduti, Mey, and Oliehoek [23] prove formally that the value loss is bounded by the approximation error. Therefore, obtaining accurate approximations is crucial for the effectiveness of the entire approach.

Even though influence approximation has been successfully applied in a variety of benchmarks, showing improvements in terms of planning and RL performance [14, 15], there exists no systematic analysis of how difficult the influence learning task is in complex scenarios and which learning methods are most effective for

[1]Note that such a collection of local problems is subject to the prize of anarchy. That is, it is not guaranteed to lead to optimal system level behavior in terms of total rewards. Nevertheless, it may allow us to deal with a complex optimization system in a tractable way, which is one of the core motivations that sparked the interest in multiagent systems [18].

this purpose. In addition, maintaining accurate approximations might become particularly difficult for long-horizon tasks. In fact, methods for sequence modeling often struggle with capturing long-term dependencies [24, 25]. Even when advanced techniques prove effective, they might not be computationally feasible, as training could require running costly long simulations. To address this gap and understand for which problems IBA can be successfully applied, we investigate the learning of accurate influence models across various realistic scenarios. We examine which learning aspects are crucial for this purpose. Thus, our aim is not to propose a novel algorithmic approach. Instead, we seek to empirically evaluate existing learning methods for approximating the influence in scenarios that may present challenges, such as complex dynamics and long-horizon problems. As key contributions of our work we show that:

- Approximating the influence might typically be an easy learning task, even for complex and large systems. In fact, small recurrent and temporal convolutional neural networks can learn accurate influence approximations across the investigated benchmarks.
- Learning models can be trained on short-horizon sequences and then deployed to approximate the influence over much longer horizons while maintaining high accuracy. Additionally, we propose a method to estimate the accuracy of the models over long horizons using the short training sequences.

In this way, we give positive evidence that influence-based abstraction might provide a feasible approach to dealing with decision making in certain real-life complex systems.

## 4.2. Background

Here we give a concise introduction to decision making problems formalized as partially observable Markov decision processes (POMDPs), as well as influence-based abstraction.

### 4.2.1. Factored POMDPs and best response problems

Formally, a POMDP [6] is a tuple $\mathcal{M} = (S, A, T, R, \Omega, O, b^0, h)$, where $S$ is the finite state space of the environment, $A$ is the finite space of available actions, $\Omega$ the observation space, and $h$ the horizon of the problem. Note that, assuming finite state and action space, we restrict our work to discrete state and action variables. Initially, a state $s^0 \in S$ is drawn from the initial distribution $s^0 \sim b^0$ [2]. At any discrete time step $t \geq 0$, the agent chooses an action $a^t \in A$, and the state changes according to the distribution $s^{t+1} \sim T(\cdot \mid s^t, a^t)$. The agent then receives a reward modeled as $r^t = R(s^t, a^t)$ and an observation $o^{t+1} \sim O(\cdot \mid s^{t+1}, a^t)$. A policy $\pi$ encodes the agent's behavior, mapping the action-observation histories $h^t = (a^0, o^1, \dots, a^{t-1}, o^t)$

[2]To ease the notation, we use small letters to denote both random variables and their realizations. Capital letters denote sets and functions. For instance, $s^0 \sim b^0$ denotes that the random variable representing the state at time 0 is distributed according to $b^0$.

into probability distributions over the action space, i.e. $\pi(h^t) \in \Delta(A)$. The goal of the agent is to optimize the expected cumulative reward for employing a policy $\pi$ given the action-observation history $h^t$: $V^\pi(h^t) = \mathbb{E}\left[\sum_{k=t}^{h} r^k \mid \pi, h^t\right]$. The policy achieving the maximum value $V^*$ is called the optimal policy and is denoted by $\pi^*$. We focus on specific domains where the state space $S$ can be decomposed into state variables or *factors*, known as factored POMDPs [26].

In line with the concept of joint equilibrium-based search for policies [27], we adopt the perspective of a protagonist agent $i$ in problems with multiple interacting agents. This approach is commonly used to solve multiagent problems, as many solution methods rely on finding the best response policies for individual agents [27–29]. Specifically, by fixing the policies of all other agents $\pi_{-i}$, the best response problem for agent $i$ can be formulated as a factored POMDP where the actions of the other agents $a_{-i}$ are included into the state space as factors. For the rest of the paper, we will omit the subscript $i$ and assume to address a best response problem for a single agent modeled as a factored POMDP and denoted as $\mathcal{M}_{\text{global}} = (S, A, T, R, \Omega, O, b^0, h)$.

### 4.2.2. A TRAFFIC EXAMPLE

We will use as a running example a traffic light control problem represented in Figure 4.1. A protagonist agent manages the traffic light at intersection 1 within a large road network. The goal of the agent is to minimize traffic congestion at

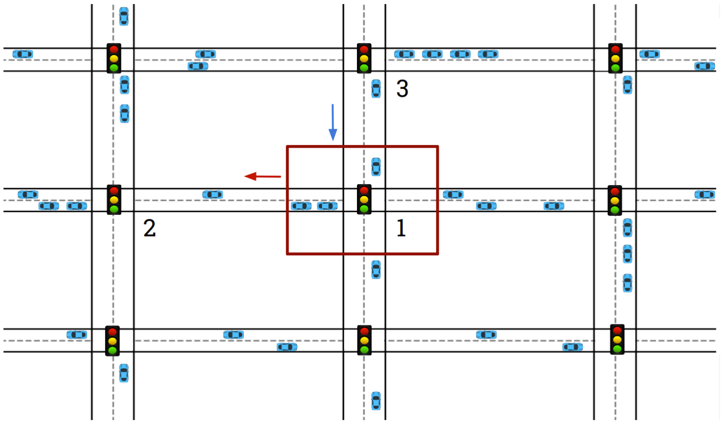


Figure 4.1.: Traffic example. The red square delimits the road segments included in the local model for the traffic light agent 1. The blue arrow represents the effect of the non-local part of the network on the local model and the red arrow the effect of the local traffic on the external part of the system.

the local intersection by leveraging the information from traffic observations from the surrounding area that is delimited by the red box. This traffic example can be represented as a factored POMDP where the state of the system is represented by the variables that measure the traffic levels at different road stretches as, for instance, the variables $s_{n\downarrow}$, $s_{w\leftarrow}$, $s_{src}$ highlighted in Figure 4.2(a). Factored POMDP can be more compactly represented by exploiting conditional independence between factors. Specifically, the transitions and observation probabilities can be represented by specific forms of Bayesian networks, the two-stage dynamic Bayesian networks (2DBNs) [30], as illustrated in Figure 4.2(b). Achieving optimal control at intersection

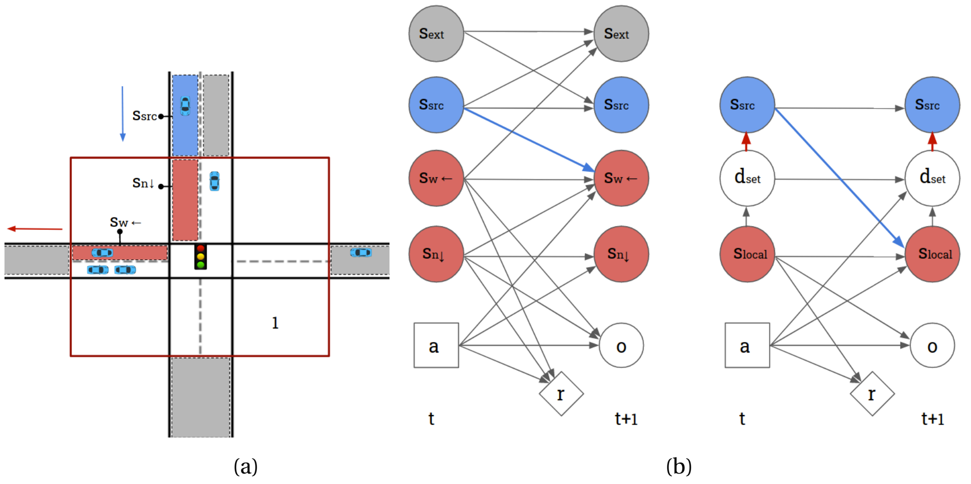


Figure 4.2.: Local model for intersection 1 of the traffic network example. (a) graphical representation of the local state variables and dependencies on external factors; (b) 2DBN representations of the local and external state variables (left) and the abstract I-ALM (right).

1 does not require simulating all the state variables. For instance, the decisions made at intersection 3 affect indirectly the observations of the agent through the car inflow from the north end (blue arrow). By abstracting away those factors that have only indirect effects on the local intersection, a smaller model can be constructed, as depicted in the red box Figure 4.2(a). In particular, the local factors highlighted in red, $s_{n\downarrow}$, $s_{w\leftarrow}$, represent the incoming and outgoing traffic flows from the north and west ends, respectively. The north end inflow denoted by $s_{src}$ and depicted in blue is a so-called *influence source* (to be defined formally alter) as it influences directly the local traffic measured by $s_{n\downarrow}$. The influence source captures the only information about the external portion of the system necessary to define the local transitions for the north end traffic $s_{n\downarrow}$.

### 4.2.3. Influence-based abstraction

As we saw in the example, the factorization of the state space allows to leverage the conditional independence between factors. By abstracting away factors that do not directly affect the agent's reward and observations, we can construct a much smaller *local model*. For instance, in the traffic example the state variables that measure the traffic outside of the area delimited by the red box can be discarded. Thus, the state of the environment can be considered as consisting of three components defined as follow.

**Definition 13.** The state $s$ can be decomposed as $s = (s_{\text{ext}}, s_{\text{src}}, s_{\text{local}})$, where $s_{\text{local}}$ represents the *local factors* retained in the model. $s_{src}$ denotes all the external state variables that directly influence the local factors and are called the *influence sources*. All the remaining factors form an external portion of the state, $s_{\text{ext}}$.

In Figure 4.2(a), the local factors are depicted in red and the influence source that affects directly the local inflow of cars from the north end is depicted in blue. All the factors that do not affect directly the local traffic are depicted in grey and form $s_{\text{ext}}$. To reduce the complexity of the problem, we want to construct the local model, corresponding to the red area in Figure 4.2(a), which therefore only contains the local factors. The 2DBN in the left-hand side of Figure 4.2(b) represents the factorization of the global model state into local factors, influence source and external factors. Note that only the local factors affect the agent's reward and observations. The blue arrow highlight the dependency of the local factor from the external influence sources.

In general, the transitions of the local factors depend on the influence sources according to the probability $\mathbb{P}(s_{\text{loc}}^{t+1} \mid s_{\text{loc}}^t, s_{\text{src}}^t, a^t)$, which are not part of the local model. For instance, the distribution of $s_{\text{n}\downarrow}$ depends on the north traffic inflow represented by $s_{\text{src}}$. One possible solution is to condition on history. Specifically, we can use the local history to predict the influence source distribution $\mathbb{P}(s_{\text{src}}^t \mid s_{\text{loc}}^0, a^0, \dots, s_{\text{loc}}^t)$. Interestingly, however, it may not be necessary to condition on the full local state-action history. Oliehoek, Witwicki, and Kaelbling [19] show that one can retain the history of only a subset of variables, the *d-set*. The d-set is defined as a set of local variables that d-separates the local factors from the external factors, according to the definition of d-separation for causal graphs in a 2DBN [31, 32]. Specifically, the influence sources at time $t$, $s_{\text{src}}^t$, are conditionally independent on the local state and action histories given the d-set, i.e. $\mathbb{P}(s_{\text{src}}^t \mid d_{\text{set}}^t, s_{\text{loc}}^0, a^0, \dots, s_{\text{loc}}^t) = \mathbb{P}(s_{\text{src}}^t \mid d_{\text{set}}^t)$. Thus, the d-set includes precisely those local factors and actions necessary to predict the influence sources.

In the traffic scenario, the history of the outgoing local traffic measured at the west end $s_{\text{w}\leftarrow}$ affects the future traffic volumes measured by the influence source $s_{\text{src}}$ and thus is part of the d-set. To predict the influence sources, we augment the local state with the local history of west end outflow $s_{\text{w}\leftarrow}$. For the sake of illustration, we will assume that the d-set consists only of those factors, i.e., $d_{\text{set}}^t = (s_{\text{w}\leftarrow}^0, \dots, s_{\text{w}\leftarrow}^t)$. In the left-hand side of Figure 4.2(b), the red arrow represents the influence, that is the distribution of the influence source $s_{\text{src}}$ given the $d_{\text{set}}$, $I(s_{\text{src}}^t \mid d_{\text{set}}^t) \triangleq \mathbb{P}(s_{\text{src}}^t \mid s_{\text{w}\leftarrow}^0, \dots, s_{\text{w}\leftarrow}^t)$. We can now provide the formal definition.

**Definition 14** (Influence)**.** The influence $I$ is the conditional probability distribution of the influence sources given the d-set at time $t$, $I(s^t_{\text{src}} \mid d^t_{\text{set}}) \triangleq \mathbb{P}(s^t_{src} \mid d^t_{\text{set}})$, defined for any time step $t < h$.

With these notions, we can define a local model as a POMDP.

**Definition 15** (Influence-augmented local model)**.** An *influence-augmented local model* (IALM) $\mathcal{M}_{\text{local}} = (S_{\text{augm}}, A, T_{\text{local}}, R, \Omega, O, b^0, h)$, corresponds to the factored POMDP where the augmented state space $S_{\text{augm}}$ comprises the local factors and the d-sets, i.e., $(s_{\text{local}}, d_{\text{set}}) \in S_{\text{augm}}$. $A$ is the action space of the agent. The local transitions $T_{\text{local}}$, which model the distributions of the local factors, are defined through the influence $I$ by marginalization over all the possible influence sources $s_{\text{src}}$. Precisely,

$$T_{\text{local}}(s^{t+1}_{\text{local}} \mid s^t_{\text{local}}, d^t_{\text{set}}, a^t) = \sum_{s^t_{\text{src}}} \mathbb{P}(s^{t+1}_{\text{local}} \mid s^t_{\text{local}}, s^t_{\text{src}}, a^t) I(s^t_{\text{src}} \mid d^t_{\text{set}}) \tag{4.1}$$

Note that we assume that the local factors encompass all the variables that directly affect the agent's reward and observations. As a consequence, the reward $R$ and observation $O$ are the same functions as those defined for the global model.

Assuming that we can derive the exact influence $I$, the I-ALM $\mathcal{M}_{\text{local}}$ defines a problem equivalent to the global model $\mathcal{M}_{\text{global}}$. That is, the two problems share the same optimal policies. Consequently, solving the smaller I-ALM yields an optimal policy $\pi^*$, which also maximizes the value of the original global problem. We refer to Oliehoek, Witwicki, and Kaelbling [19] for a formal proof of this equivalence and an extensive discussion of the IBA concepts introduced in this section. However, deriving the influence for any possible d-set requires solving an exponential number of inference problems in the horizon $h$. For instance, to define the influence for the local traffic model, we would need to compute a number of conditional probability distributions of the order of $(\max s_{\text{w}\leftarrow})^h$, which corresponds to the cardinality of the d-set. This task is practically infeasible even for small values of the variables. This challenge leads to the idea of learning approximations of the influence $\hat{I}$.

### 4.2.4. APPROXIMATE INFLUENCE-BASED ABSTRACTION

Given a representation $\hat{I}$, an *approximate* influence-augmented local model, a $\hat{I}$-ALM is defined following the same structure of Definition 15 as the factored POMDP $\hat{\mathcal{M}}_{\text{local}} = (S_{\text{augm}}, A, \hat{T}_{\text{local}}, R, \Omega, O, b^0, h)$, with local transitions $\hat{T}_{\text{local}}$ defined by replacing the exact influence $I$ in (4.1) with the approximate influence $\hat{I}$. To construct an $\hat{I}$-IALM, we need to learn $\hat{I}$. For this task, we first simulate $n$ trajectories of influence sources and d-sets from the global simulator to form the training set $\mathcal{D}_h = \{(d^0_{\text{set}}, s^0_{\text{src}})_i, \ldots, (d^h_{\text{set}}, s^h_{\text{src}})_i\}_{i=1,\ldots,n}$. Now, we define the learning task.

**Definition 16** (Influence learning task)**.** Given a training set of influence sources and d-sets trajectories $\mathcal{D}_h$, the influence learning task consists of learning a predictor for the conditional distribution of the influence sources $s^t_{\text{src}}$ given the local history in the d-set $d^t_{\text{set}}$, $\hat{I}(s^t_{\text{src}} \mid d^t_{\text{set}})$ for any $t = 0, \ldots, h-1$.

A probabilistic model parameterized by $\theta$, for example a neural network, can be used for this sequence modelling task to approximate the influence as $\hat{I}(\theta) = \hat{I}(s^t_{\text{src}} \mid d^t_{\text{set}}; \theta)$. The learning model is trained to solve the optimization problem formulated as

$$\theta^\star = \arg\min_\theta \mathbb{E}_{\mathscr{D}_h} \left[ \text{CE}(I(\cdot \mid d^t_{\text{set}}) \| \hat{I}(\cdot \mid d^t_{\text{set}}; \theta)) \right] \tag{4.2}$$

where the expectation corresponds to the average cross entropy loss between the target influence and the approximation model.

As mentioned in the introduction, the advantage of this approach lies in the possibility of using a small sample $\mathscr{D}$ from the global simulator to build the local lightweight simulator $\hat{I}$-ALM, which enables more efficient sampling and accelerate the solutions search. Moreover, Congeduti, Mey, and Oliehoek [23] provide additional support for the approximate abstraction by demonstrating that the value loss for solving the sequential decision-making problem defined by the $\hat{I}$-ALM is bounded by the worst-case KL divergence error of the influence predictions over all the possible d-sets. This bound guarantees that any influence $\hat{I}$ that minimizes the mean cross entropy loss according to (4.2)–and thus the mean KL divergence–is aligned with the objective of minimizing the value loss.

### 4.2.5. STATE DECOMPOSITION

In our work, we assume that the decomposition into local and external factors is an engineering choice and thus given, with the constraint that the local factors include all variables that directly affect the reward and the observations. This ensures that reward and observations remain well-defined at local level [19]. Ideally, the local model should be as small as possible to enable simulation speedups. However, in some cases a larger local model can ease the problem of predicting the influence sources. This may be viewed as a potential limitation, since finding a good trade-off between the information included in the local model and the model size may not be trivial. However, in practice, we have generally found it relatively straightforward to identify sets of local variables that yield good performance [17]. Although we recognize that different decompositions may lead to different outcomes, any choice determines a set of influence sources and d-set, and thus defines an influence learning problem. Assessing the impact of different decomposition on the complexity of the resulting influence learning problem is quite a challenging task. Therefore, a comprehensive analysis of the effect and efficiency of different choices is beyond the scope of this paper. On the other hand, given a local model, identifying an optimal d-set is relatively straightforward: we select a minimal d-set, as this is also the best choice for the set of input features to the learning models. Using a larger (still d-separating) set of local variables will not improve the accuracy of the learning models. Instead, it would increase the dimensionality of the input of the learning problem, generally making learning more difficult.

## 4.3. Empirical study of influence learning

The influence learning problem is a sequence prediction task that corresponds to modeling the temporal dependencies between the local and the external factors governed by an underlying 2DBN. This structure differs substantially from the benchmark domains typically used to test sequence modeling methods, such as speech recognition, machine translation, audio classification, music generation, image-video caption generation [33, 34]. Thus, this learning problem warrants separate investigation. Based on the intuition that humans are generally quite successful at reasoning about parts of complex systems in isolation, we hypothesize that the local structure of the problems may make the task of learning the influence quite feasible and manageable, even with relatively low-complexity methods for sequence predictions.

To make an impact on real problems with IBA, we need to address two important aspects. First, real-world tasks are often large and complex, which means that we need to explore if approximating the influence is feasible for such problems. Additionally, many real-world tasks often involve systems, such as traffic or networks, that operate continuously, which means that the learning models should work well for very long (or infinite) horizons. For the entire IBA approach to be effective, we need to have sufficiently accurate influence approximations in these situations. Therefore our investigation focuses on two key aspects:

1. Testing the hypothesis that even for large and complex systems, the learning problem may turn into a manageable task that does not require large or complex neural networks.
2. Investigating to what extent the representations learned from trajectories with a short training horizon $h_{\text{train}}$ can approximate well the influence over a much longer deployment horizon $h_{\text{deploy}} \gg h_{\text{train}}$.

### 4.3.1. Simulation domains

Our experimental setup includes a range of realistic simulation domains, covering diverse situations in terms of number of influence sources to predict, level of 'uncertainty' of their distributions, strength of the dependency between local variables and influence sources, dependency over past time steps, and problem horizons. This variety leads to different characteristics of the learning tasks. Our aim is to determine if this has an impact on the performance of the learning models. We consider four simulation domains designed to test the efficacy of local influence proxies on realistic tasks:

- **Microgrid (MG)**: A realistic application of power system management in the energy engineering sector. Similar to Vazquez-Canteli *et al.* [35], in this environment a hundred prosumers (units that both produce and consume electrical power) interact by trading energy to minimize the costs of energy bought from an external grid while meeting the internal power demand. This use case closely emulates problems of practical importance in power engineering, particularly in modern contexts involving distributed energy prosumers, strategies for managing energy storage systems, and recent efforts toward achieving net-zero, positive-energy districts.

- **Traffic grid (TG)**: A traffic light control problem that models the interactions between cars and traffic lights in a road network, as briefly introduced in Section 4.2.2. This scenario features many external factors which exert a direct influence on the local model. Hence, the influence learning problem corresponds to a high-dimensional prediction task. This transportation network use case serves as a surrogate for real-world challenges, such as optimizing traffic light operations to minimize queue times and reduce congestion.
- **System admin (SA)**: A multiagent version of the system administrator domain described in Poupart and Boutilier [36]. This environment reproduces a realistic challenge in the information technology domain: a team of system administrators has to manage a network of potentially faulty machines by deciding which machine to reboot.
- **Grab a chair (GC)**: A simple gaming environment introduced by He, Suau de Castro, and Oliehoek [13], where a group of agents competes to obtain one of the available chairs. This environment is a simplified version of the system admin serves as a control scenario for proof-of-concept experiments and preliminary investigations.

We refer the interested reader to Appendix C.2 for a detailed description and visualization of the simulation domains.

### 4.3.2. Comparison of learning models

We evaluate the performance of different classes of learning models with various degree of complexity [37], measured as the network size or number of learnable parameters on the influence learning task.

**Models and benchmark domains used for this experiment.** Many successful sequence prediction methods for sequential predictions are based on neural network with recurrent and temporal convolution components [22, 33, 38]. We evaluate several classes of models including recurrent models such as long short-term memory (LSTM) [39] and gated recurrent unit (GRU) [40], as well as temporal convolution-based models like temporal convolutional network (TCN) [41] and fully convolutional network (FullyConv) [42]. To compare these models with a simpler non-recurrent baseline, we assess a one-layer fully connected linear network, a logistic regression (LogReg) model. We exclude more complex and larger models, such as transformers [43, 44], as we assume, and demonstrate, that small and simple models can provide accurate influence representations. For each model class, we compare different sizes, ranging from lightweight networks with less than 100 parameters scaling up to the size a one-layer linear neural network counting more than one million parameters. The hyperparameters defining the network architectures have been chosen to ensure a fair comparison between learning models with the same network size. Since our goal is to investigate whether relatively small and simple models can achieve satisfactory performance, we focus on models with a small number of layers. This choice is also motivated by the fact that deeper networks typically require a larger amount of global simulator trajectories to be trained effectively [45]. Moreover, the choice of the network depth depends on the model class. Specifically, convolutional-based models are known to benefit from deeper architectures to achieve better performance and to ensure full receptive field coverage [46, 47]. For the details of the architectures and sizes used in the different domains, we refer to Table C.6, C.7, and C.8 in Appendix C. The models are evaluated over three domains: microgrid, system admin, and traffic grid. For each domain, we choose the scenario features, including the number of interacting agents $N$, the protagonist agent $i$, the policies of the external agents $\pi_{-i}$, the problem horizon $h$, the initial distribution $b^0$ and other domain-specific parameters that define the transition and reward functions. These scenarios are designed to cover a range of diverse characteristics in terms of problem size, time length, prediction dimension and stochasticity for the influence learning task. Specifically, the microgrid represents a large scenario where 100 agents interact within a power grid; the traffic grid scenario is a real-world traffic control application that turns into a high dimensional learning task; and the influence learning problem for the system admin simulator is a long-horizon forecasting problem.

**Experimental setup.** We use a random exploratory policy for the local agent $\pi_i^{\text{Exp}}$ which affects only the data distribution of the training set. We collect $n = 500$ trajectories of influence sources and d-sets from the global simulator to form the training set $\mathcal{D}_h$. The different models are trained to approximate the influence as $\hat{I}$, using the mean cross entropy as the training loss. We employ $h$ independent logistic regression models, each one representing the influence $\hat{I}(s^t_{\text{src}} \mid d^t_{\text{set}})$ for a specific time step $t \in \{0, \ldots, h-1\}$. We adopt standard optimization techniques, including the ADAM optimization algorithm, linear decay of the learning rate and grid search over the space of initial and final learning rates. For each scenario, a fixed number of epochs and the batch size are selected. Details on the scenarios and hyperparameter configurations are provided in Table C.1 and C.2 in Appendix C, respectively.

**Performance metrics.** The model performance is assessed on an a test set consisting of $n_{\text{test}}$ independent global model trajectories. We use the cross entropy test error, which is computed as the mean over time and sum over the influence sources of the cross entropy estimators. Specifically, for each factor $s^t_{\text{src},j}$ of the influence sources $s^t_{\text{src}} = (s^t_{\text{src},1}, \ldots, s^t_{\text{src},J})$, an estimator of the cross entropy error $\text{CE}(I(s^t_{\text{src},j} \mid d^t_{\text{set}}) \| \hat{I}(s^t_{\text{src},j} \mid d^t_{\text{set}}))$ at time $t$ is computed over the test sample. Then, the sum over the $J$ influence sources provides the error for a given time step $t$ according to the following formula:

$$\text{CE}^t(I, \hat{I}) = \sum_{j=1}^{J} \left[ -\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \ln \hat{I}(s^t_{\text{src},j,i} \mid d^t_{\text{set},i}) \right] \qquad (4.3)$$

With a slight abuse of notation, we denote both the probability distribution of individual influence sources as $\hat{I}(s^t_{\text{src},j} \mid d^t_{\text{set}})$ and the joint distribution as $\hat{I}(s^t_{src} \mid d^t_{\text{set}})$. The cross entropy error is obtained by averaging the errors defined per time step in (4.3) as follows:

$$\mathrm{e}(h) = \frac{1}{h}\sum_{t=0}^{h-1} \mathrm{CE}^t(I, \hat{I}) \tag{4.4}$$

We also measure the wall-clock training times (WCTTs) to assess which model provides the best trade-off between accuracy and training time.

### 4.3.3. Generalization beyond the training horizon

In many real-world applications, an approximate model $\hat{I}$ should perform well over long horizons. Examples of problems that require long-term simulations include long horizon planning [48, 49], episodic reinforcement learning [50] and tasks characterized by sparse reward signals [51]. Although it might be feasible to gather a sufficient number of trajectories from the global simulator to approximate well the influence, this may no longer be the case when dealing with long horizon problems as long-term dependencies add further complexity to the learning task.

Our approach to address this issue is to leverage learning models to generalize the influence representations beyond the training horizon. This intuition is grounded in the ergodic theory for MDPs [52–54]. Under ergodic assumptions, the Markov dynamics induced by a joint policy $\pi = (\pi_1, \ldots, \pi_N)$ converge to a unique stationary distribution independently of the initial distribution. The mixing time $t_{\text{mix}}$, refers to the number of time steps required for the state distribution to approximate the stationary distribution within a specified tolerance. Such mixing time is determined solely by intrinsic properties of the system and is independent of the initial conditions. This argument supports the concept of a *stationary influence* as the limit of the conditional distributions of the sequence $\{I(y^t_{\text{src}} \mid d^t_{\text{set}})\}_t$. In other words, when the system has sufficiently mixed, the influence approaches a time-independent function, representing the distribution of the influence sources at equilibrium. In this scenario, global trajectories with horizon $h_{\text{train}} > t_{\text{mix}}$ contain sufficient information to learn the stationary influence. Therefore, if the mixing time is sufficiently short, learning models can capture the stationary influence using short training sequences and use it to predict effectively the influence over (indefinitely) longer deployment horizons $h_{\text{deploy}} \gg h_{\text{train}}$.

How to choose an appropriate training horizon for this task remains an open question. We expect predictions to improve as the training horizon $h_{\text{train}}$ approaches the deployment horizon $h_{\text{deploy}}$. However, we aim to limit the computational effort for running long global simulations. Therefore, the training horizon must offer a good trade-off between the quality of approximations and the length of the training sequences. Specifically, we seek an optimal training horizon $h^*_{\text{train}}$ as the minimum number of time steps required to guarantee accurate predictions across the entire deployment horizon. Intuitively, the optimal horizon $h^*_{\text{train}}$ depends on the system's mixing properties and the ability of the learning models to capture the stationary distribution. Specifically, we verify that it corresponds to the sum of the mixing time

$t_{\text{mix}}$ and a few additional time steps needed for the learning models to generalize the experience encountered after mixing.

**Models and benchmark domains used for this experiment.** LSTM and FullyConv networks are employed for the experiments, as they achieve the best performance in the experiments of the previous section on the influence learning tasks. The architectural hyperparameters are selected to ensure that the learning models share the same size. Tables C.4 and C.5 in Appendix C report the optimization and network architecture details, respectively. We use the grab a chair and traffic grid domains as described in Section 4.3.1. Grab a chair serves as controlled environment where a known stationary influence is reached after a few time steps of the mixing time. Specifically, we consider two different scenarios: the first with $N=4$ interacting agents denoted as GC4, and the second with $N=11$ agents denoted as GC11. We assume that the agents are ordered by index and each external agent $2,3\dots$ copies the action of the preceding agent at previous time step (see Figure C.5 in Appendix C.2). For instance, agent 3 at time $t$ copies the last action of agent 2, which is, in turn, the action of the local agent 1 at time $t-2$.In general, for any agent $i$, $a_i^t = a_1^{t-i+1}$. Note that the external agent's policies are arbitrary features of the experimental domains and, as such, can be chosen to induce various influence learning tasks. This ad-hoc choice ensures that after the mixing time $t_{\text{mix}} = N-1$ the system converges to a known stationary distribution determined by the local agent policy, i.e. $a_i^t \sim \pi_1^{\text{Exp}}$ for $t \geq N-1$. As a result, the influence is a time independent deterministic function of the last $N$ local actions determined by the following formula

$$I\left(a_2^t, a_N^t \mid a_1^0, \dots, a_1^{t-1}\right) = \mathbb{P}\left(a_2^t, a_N^t \mid a_1^{t-1}, a_1^{t-N+1}\right) = \delta_{a_1^{t-1}}\left(a_2^t\right)\delta_{a_1^{t-N+1}}\left(a_N^t\right) \qquad \text{for } t \geq N-1 \tag{4.5}$$

where $\delta_a$ denotes the Dirac distribution centered on the action $a$. Such explicit expression for the stationary influence allows us to analyze the model learning for the different training horizons. We also use the traffic grid domain to validate our arguments in a more realistic environment where no prior knowledge of stationarity and mixing time is available. Details on the scenarios, training and deployment horizons can be found in Table C.3 in Appendix C.

**Experimental setup.** To empirically validate these arguments, we proceed as follows. We consider a set of $K$ training horizons $H = \{h_1, \dots, h_K\}$ with $h_1 \leq \dots \leq h_K$. We employ a random exploratory policy $\pi^{\text{Exp}}$ for the local agent and collect $n$ global trajectories of d-sets and influence sources with horizon $h_K$. Then, for each model class, we train $K$ learning models, such that the $k$-th model uses the training set $\mathscr{D}_{h_k}$, which consists of trajectories up to horizon $h_k$, to learn the approximate influence $\hat{I}_k(\theta)$. To assess the generalization ability, we test the models over longer trajectories. For this purpose, we collect an independent test sample of trajectories with horizon $h_{\text{deploy}} \gg h_K$ denoted by $\mathscr{D}_{h_{\text{deploy}}}$.

**Deployment, test and test-tail error.** We define the function $e^k(h)$ to represent the generalization error of the $k$-th model when tested over $h$ deployment steps, as given by the following formula:

$$\mathrm{e}^k(h) = \frac{1}{h}\sum_{t=0}^{h-1} \mathrm{CE}^t(I, \hat{I}_k(\theta)) \tag{4.6}$$

where the cross entropy term for time step t is defined in (4.3). Note that for $h = h_{\text{deploy}}$, the error $\mathrm{e}^k(h_{\text{deploy}})$ corresponds to the average error over $h_{\text{deploy}}$ time steps. This error measures how well (on average) the $k$-th model generalizes the influence over the entire deployment horizon. While for $h = h_k$, the error $\mathrm{e}^k(h_k)$ represents the average error of the $k$-th model over $h_{\text{train}} = h_k$ time steps. We refer to these errors as the *deployment error* and the *test error*, respectively, and denote them as follows:

$$\mathrm{e}^k_{\text{deploy}} := \mathrm{e}^k(h_{\text{deploy}}) \qquad\qquad \mathrm{e}^k_{\text{test}} := \mathrm{e}^k(h_k) \tag{4.7}$$

A learning model that shows good long-term performance should maintain the deployment error close to the test error. Additionally, increasing the training horizon should not significantly reduce the deployment error.

We are also interested in investigating how to estimate the deployment error from the short training trajectories. While the test error might intuitively seem a good candidate estimator, it is significantly affected by the error terms before the system has mixed. To illustrate this, we distinguish the cross entropy error terms in (4.6) into two time scales: the short time scale of the mixing time and the long time scale of the deployment horizon. This distinction allows us to break the error function into two terms as follows:

$$\mathrm{e}^k(h) = \frac{1}{h}\left[\sum_{t=0}^{t_{\text{mix}}-1} \mathrm{CE}^t(I, \hat{I}_k(\theta)) + \sum_{t=t_{\text{mix}}}^{h} \mathrm{CE}^t(I, \hat{I}_k(\theta))\right] \tag{4.8}$$

The first term accounts for the short time errors over the influence before mixing and the second for long time errors which represent the model's performance over the stationary influence. Given that $h_{\text{deploy}} \gg t_{\text{mix}}$, the deployment error is mainly affected by the second term of (4.8), while the contribution of the short time errors is negligible. Conversely, when the training horizon $h_{\text{train}}$ is close to the mixing time, the first time errors predominantly determine the test error. These errors do not reflect the model's performance on the stationary influence, causing the test error to deviate from the deployment error.

The key idea for deriving a more accurate estimator is to neglect the errors over the initial time steps. However, the appropriate number of time steps to discard depends on the exact mixing time of the system, which is generally unknown and difficult to estimate. Therefore, we consider different increasing values of the training horizon $h_k$ to test the hypothesis that for certain $h_k > t_{\text{mix}}$, the 'tails' of the error represented in (4.8) are close enough to the deployment error. Specifically, we introduce the *test-tail error*, which is computed as the average error over a window of $l$ time steps, according to the following formula:

$$\mathrm{e}_{\text{tail}}^{k} = \frac{1}{l} \sum_{t=h_k+1}^{h_k+l} \mathrm{CE}(I, \hat{I}_k(\theta)) \tag{4.9}$$

In other words, for training horizon $h_k$, $l$ additional time steps are collected in the test set $\{((d_{\text{set}}^{h_k+1}, s_{\text{src}}^{h_k+1}), \ldots, (d_{\text{set}}^{h_k+l}, s_{\text{src}}^{h_k+l}))_i\} \subset \mathcal{D}_{h_{\text{deploy}}}$ and used to compute the test-tail error. We empirically show that in our scenarios the error $\mathrm{e}_{\text{tail}}$ offers a good estimate of $\mathrm{e}_{\text{deploy}}$ for $l = 1$, and can thus be used to assess the quality of the model predictions over the deployment horizon. However, we recognize that increasing the number of time steps, i.e. $l > 1$ might prevent disruptive effects of anomalies and give more stability to the test-tail error. Additionally, we show how this error can be used to search for an optimal training horizon $h_{\text{train}}^{*}$.

## 4.4. Results and discussion

Here, we present the empirical results for the two sections of the experiments presented.

### 4.4.1. Comparison of learning models

We start by analyzing the test error of different model classes and sizes as a function of the number of epochs for the three domains. Together with the training curves, we look at the final test errors and wall-clock training times. To compare the model classes, we select one network size for each class from the Pareto optimal solutions [55] of the bi-objective problem of minimizing test error and training time. Finally, we look at the progression of the test error over training time to compare the learning speed and accuracy of the model classes.

**Effect of model class and size.** Figure 4.3 depicts the progression of the test errors, computed according to (4.4) over the training epochs of the four classes of models (rows) in the three problem domains (columns). For each class and benchmark domain the training curves for different network sizes are illustrated as average test error with standard error over ten iterations of the experiment. The final test errors and training times are reported in Table 4.1 as averages over the ten iteration of the experiment. The corresponding standard errors are reported for the cross entropy errors but not for the wall-clock training times, as they are negligible.

For the microgrid, the training curves and final test errors show no significant advantages from larger model sizes, except in the case of the TCN model class. In fact, the recurrent-based models, such as LSTM and GRU, do not benefit from increased sizes. Conversely, increasing the TCN size from fewer than 100 parameters to 15,000 parameters significantly improves the performance, reducing the test error from 4.11 to 3.85. For the FullyConv model, a bigger size leads to a slight performance improvement but triples the training time from 0.46 to 1.55 seconds. These results align with the observation that temporal convolutional networks typically benefit from deeper architectures and thus require more parameters than

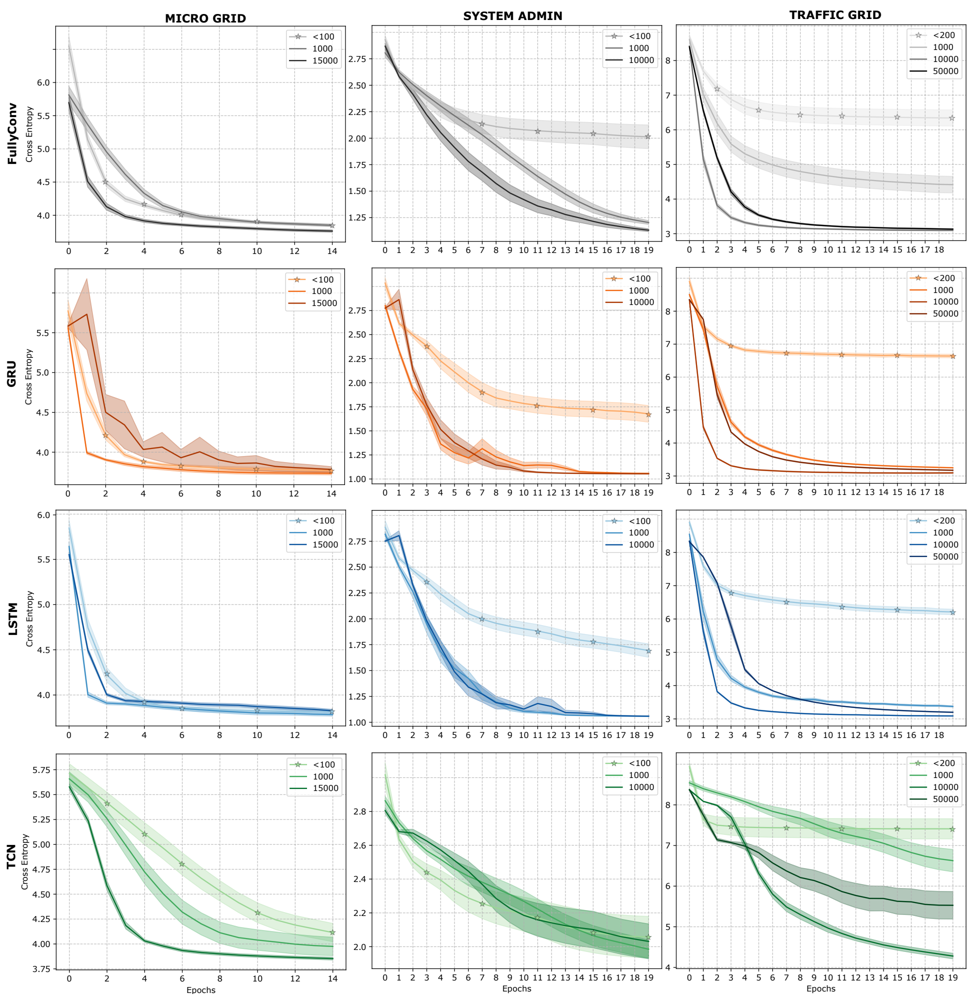


Figure 4.3.: Effect of model size and class on the different benchmark problems. Each panel depicts the cross entropy test error over the training epochs with standard error over ten iterations for different network sizes. The columns represent the three different test environments and the rows the classes of the learning model.

recurrent networks [33]. Especially for long sequences, convolutional networks need several layers to reach a full receptive field, that is to ensure that the entire history-length of the input sequence is processed to make predictions. Despite the higher number of parameters, the TCN model have typically low training times. In fact, the TCN model with 15,000 parameters achieves accuracy to the other model classes in about the same training time. Figure 4.3 displays also the results on the system admin and traffic grid domains in the second and third columns, respectively. In contrast with the microgrid domain, very small recurrent (LSTM, GRU) and FullyConv models do not have enough capacity to accomplish the learning tasks. This is due to the higher dimensionality of the space of influence sources and d-sets in the traffic domain and the longer problem horizon for the system admin. These features increase the complexity of the two learning problems compared to the microgrid case.

| | **CE / WCTT number of parameters** | | |
|---|---|---|---|
| **models** | ≤ 100 | 1000 | 15000 |
| LSTM | 3.81 ± 0.03 / 0.98 | 3.78 ± 0.02 / 1.18 | 3.82 ± 0.03 / 2.26 |
| GRU | 3.77 ± 0.03 / 0.95 | 3.73 ± 0.03 / 1.13 | 3.78 ± 0.03 / 2.15 |
| TCN | 4.11 ± 0.03 / 0.47 | 3.98 ± 0.03 / 0.59 | 3.85 ± 0.04 / 1.31 |
| FullyConv | 3.85 ± 0.03 / 0.46 | 3.85 ± 0.03 / 0.65 | 3.76 ± 0.03 / 1.55 |
| LogReg | - | - | 3.85 ± 0.04 / 0.95 |

(a) Microgrid.

| | **CE / WCTT number of parameters** | | | |
|---|---|---|---|---|
| **models** | ≤ 200 | 1K | 10K | 50K |
| LSTM | 6.2 ± 0.4/20.1 | 3.4 ± 0.3/20.9 | 3.1 ± 0.3/25.4 | 3.2 ± 0.3/40.2 |
| GRU | 6.6 ± 0.6/21.4 | 3.3 ± 0.2/22.2 | 3.1 ± 0.3/27.5 | 3.2 ± 0.3/40.6 |
| TCN | 7.4 ± 0.2/6.0 | 6.6 ± 0.0/7.0 | 4.3 ± 0.4/8.2 | 5.5 ± 0.5/14.3 |
| FullyConv | 6.3 ± 0.2/6.5 | 4.4 ± 0.3/7.9 | 3.1 ± 0.3/10.3 | 3.1 ± 0.3/17.0 |

(b) Traffic grid. LogReg (1M parameters): 5.6 ± 0.3/35.3.

| | **CE / WCTT number of parameters** | | | |
|---|---|---|---|---|
| **models** | ≤ 100 | 1K | 10K | 3M |
| LSTM | 1.69 ± 0.04/13.7 | 1.06 ± 0.04/18.5 | 1.05 ± 0.04/32.4 | - |
| GRU | 1.68 ± 0.04/13.1 | 1.06 ± 0.04/18.3 | 1.05 ± 0.04/33.9 | - |
| TCN | 2.06 ± 0.05/3.6 | 1.98 ± 0.03/5.1 | 1.05 ± 0.04/12.7 | - |
| FullyConv | 2.01 ± 0.04/3.9 | 1.20 ± 0.05/5.2 | 1.05 ± 0.04/12.3 | - |
| LogReg | - | - | - | 1.97 ± 0.08/23.2 |

(c) System admin.

Table 4.1.: Cross entropy test error with standard error and wall-clock training time (s) for the microgrid, traffic grid and system admin computed over ten iterations.

**Pareto optimal sizes and learning speed.** Figure 4.4 shows the Pareto frontiers for the model classes (left panels) where each marker represents the test error and training time for a specific size. The dashed lines indicate the Pareto fronts for each

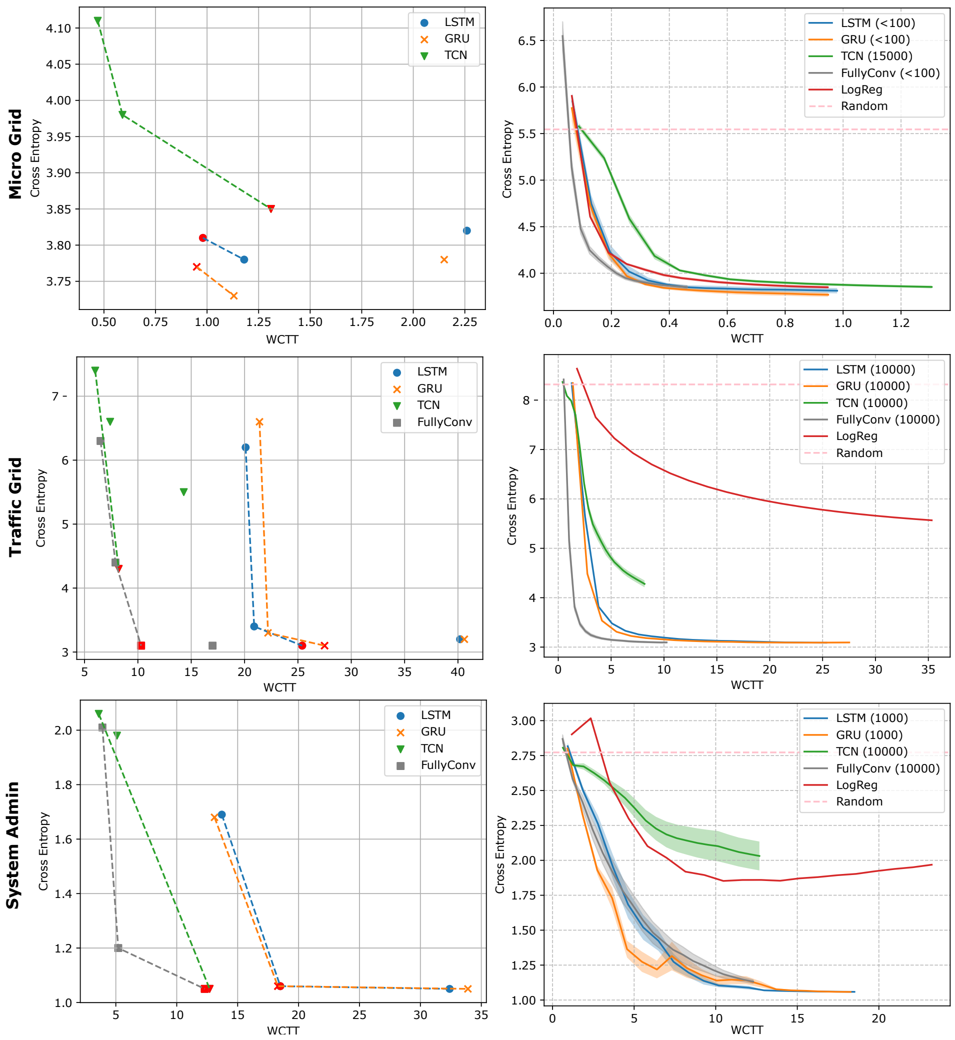


Figure 4.4.: Left panels: Pareto frontiers for the model classes LSTM, GRU, TCN and FullyConv for the cross entropy test error on the y-axis and wall-clock training time on the x-axis for the domains MigroGrid, TrafficGrid and System admin. The sizes selected are represented with red markers and correspond to the cells highlighted in Table 4.1. Right panels: cross entropy test error over wall-clock training time for the sizes of the network selected for each model class. The dashed line represents the cross entropy error of the baseline random classifier.

class, defined by the sizes that are not strictly dominated by any other in the same class. Among the Pareto optimal solutions, we select and analyze specific sizes which provides a good trade-off between error and training time. These are indicated by the red markers. On the right-hand side of Figure 4.4, we collect the test error over training time for the selected sizes and compare them with two baseline models: a random model and LogReg.

For the microgrid, the Pareto fronts show that the LSTM, GRU and FullyConv models achieve approximately the same accuracy for the different sizes. Thus, the smaller sizes are selected and highlighted in grey in Table 4.1, as they also offer the lowest training time. Conversely, the TCN model's error drops significantly with larger network size, leading to the choice of size 15,000 for that class. When comparing the training curves of the selected models, we observe similar learning speed and final accuracy across all models. Notably, the logistic regression model achieves comparable performance to the other more complex learning models. One explanation for this result is that, despite the complex nature of the realistic microgrid scenario, the corresponding influence learning task is relatively simple: a low-dimensional forecasting problem with influence sources weakly dependent on the local history. Even scaling up the microgrid to include many more units would still result in a low dimensional and simple influence learning task. This situation is representative of several complex realistic scenarios where the local model is well-decoupled from the rest of the system, with only a few external influence sources weakly affecting the local dynamics.

A different situation arises in the traffic grid and the system admin domains which induce more complex influence learning tasks in terms of dimensionality and problem horizon, respectively. In these domains, the linear regression and TCN models show their limitations. In the right panels of Figure 4.4), the red training curves show that the logistic regression fails to achieve good accuracy levels. Although the green training curves for the TCN model have not yet reached convergence, the learning is much slower compared to the FullyConv, LSTM and GRU models. However, even for these problems characterized by higher dimension or longer horizons, relatively small architectures for LSTM, GRU and FullyConv models (between 1000 and 10000 parameters) achieve good performance.

**Error over time steps.** To further analyze the limitation of LogReg model for long horizon problems, we plot in Figure 4.5 the test error per time step of the logistic regression compared to LSTM for the system admin domain. While the LSTM error remains relatively constant, the error for logistic regression increases over the time steps. This suggests that the logistic regression's ability to represent the influence diminishes with longer horizons. One possible explanation is that linear models struggle to accurately capture increasingly nonlinear relationships between local variables and influence sources as the problem horizon becomes longer.

### 4.4.2. Generalization beyond the training horizon

First, we look at the deployment errors for specific training horizons to show that it is possible to generalize the influence approximations over long deployment horizons. Then, we analyze the effect of the training horizon on the accuracy of the long-term approximations by looking at the deployment error and training curves

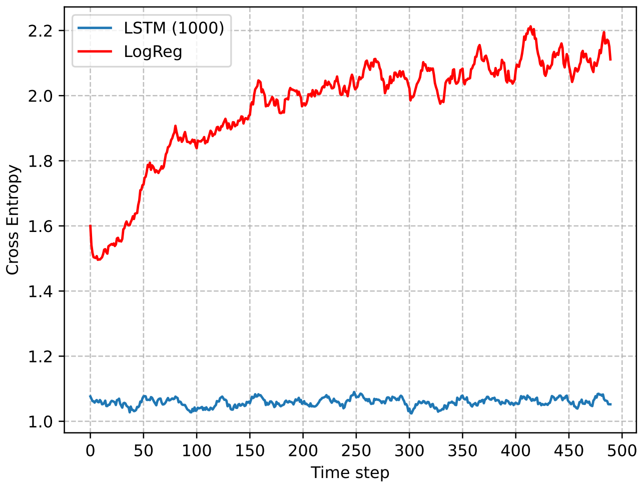


Figure 4.5.: Comparison of cross entropy test error across different time steps in the system administration domain between an LSTM model with 1000 parameters (blue) and the baseline logistic regression model (red). The curves show that the error of the logistic regression increases over time steps, while the LSTM maintains a stable error.

for different training horizons. Finally, we compare deployment, test and tail-test error for different training horizons to identify which error serves as a reliable proxy for the deployment error.

Table 4.2 shows the average deployment errors $e_{\text{deploy}}$ with corresponding standard errors for specific choices of $h_{\text{train}}$, estimated over ten iterations of the entire experiment. The performance of the models in the three scenarios is compared with the error of a random classifier. The choice of a suitable training horizon is domain-dependent: in the grab a chair scenario with 4 agents, $h_{\text{train}} = 6$ training steps are sufficient to get deployment error close to 0 while for 11 agents, the models require a longer training horizon $h_{\text{train}} = 22$. Such choices have been driven by the idea that the training horizons need to be longer than the mixing times,

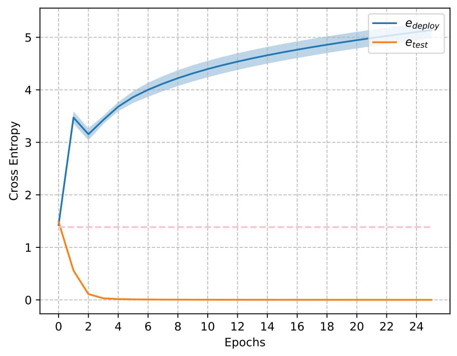
| | **learning model** | | | | |
|---|---|---|---|---|---|
| **scenario** | LSTM | FullyConv | Random | $h_{\text{train}}$ | $h_{\text{deploy}}$ |
| GC4 | $0.002 \pm 0.002$ | $0.027 \pm 0.026$ | 1.38 | 6 | 200 |
| GC11 | $0.002 \pm 0.001$ | $0.025 \pm 0.01$ | 1.38 | 22 | 200 |
| TG | $3.68 \pm 0.06$ | $3.98 \pm 0.07$ | 8.32 | 30 | 500 |

Table 4.2.: Mean and standard error of the deployment error over ten iterations.

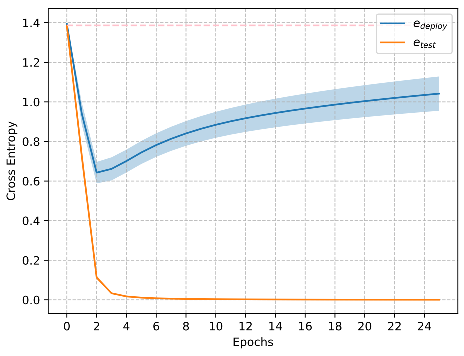
which corresponds to $t_{\text{mix}} = 3$ and $t_{\text{mix}} = 10$, respectively. The results show that a training horizon slightly longer than the mixing time ensures cross entropy errors close to zero over a much longer deployment horizon. In other words, few training time steps of experience after mixing are sufficient for the models to generalize the deterministic stationary influence over 200 deployment time steps. For the traffic grid domain, the results are less straightforward to interpret. In fact, the cross entropy error depends on the entropy of the target influence sources distributions, which are unknown. However, for $h_{\text{train}} = 30$ the average errors over 500 deployment time steps in Table 4.2 are significantly lower than the random classifier error. Also, they are quite close to the errors computed over a much shorter test horizon and reported in Table 4.1b. This leads to conclude that 30 training steps are sufficient to learn good long-term influence approximations. In summary, for every scenario considered, we found a sufficient number of training steps $h_{\text{train}}$ to learn influence approximations for much longer deployment horizon $h_{\text{deploy}}$ ensuring small deployment error. To gain a concrete and visual understanding of the predictions of the models reported in Table 4.2, we refer to the videos in Section C.3 of Appendix C.

**Optimal training horizon.** The left side panels of Figure 4.7 display the deployment errors $\mathrm{e}^{k}_{\text{deploy}}$ of the LSTM and FullyConv models computed for increasing training horizons $h_k \in H$ for the three scenarios. Notably, for GC4 and GC11 the deployment errors drop significantly when $h_{\text{train}} \geq 3$ and $h_{\text{train}} \geq 10$, respectively (see top left and central panels). Clearly, this drop correlates with the mixing times of the systems that correspond to $t_{\text{mix}} = 3$ for GC4 and $t_{\text{mix}} = 10$ for GC11. Essentially, both learning models exhibit improved accuracy as soon as some experience of the steady state is recorded in the training trajectories. However, to achieve error close to zero a few additional training time steps are needed, precisely $h_{\text{train}} \geq 6$ for CG4 and $h_{\text{train}} \geq 20$ for CG11.

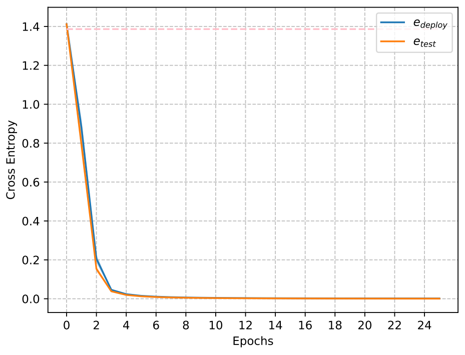
To better understand the impact of the mixing time on the model learning, we compare the deployment and test errors progression over training epochs of the LSTM for different training horizons on the CG4 domain in Figure 4.6. Before the mixing time for $h_{\text{train}} = 2$, no experience of the steady state is stored in the training set. Thus, all the predictions are solely based on the influence experienced in the first 2 time steps. As a consequence, Figure 4.6(a) shows an overfitting effect: the test error improves over the training epochs and tends quickly to zero as the approximations are very accurate for the first time steps; on the other hand, the deployment error becomes larger for increasing training epochs. After mixing, for $h_{\text{train}} = 4$ the deployment error has a significant decrease. Yet Figure 4.6(b) shows the same overfitting effect for more training epochs. The reason is that the experience of stantionary influence in the training sequences is still very limited and not sufficient for the learning model to generalize it over the deployment horizon. For $h_{\text{train}} = 6$, Figure 4.6(c) shows that the deployment error quickly tends to zero together with the test error. Moreover, no significant differences are detected for additional training time steps (see Figure 4.6(d) for $h_{\text{train}} = 8$). This motivates the choice of optimal training horizon as $h^{*}_{\text{train}} = 6$. Similarly for GC11, the training curves in Figure C.1 Appendix C show that the deployment error starts to decrease



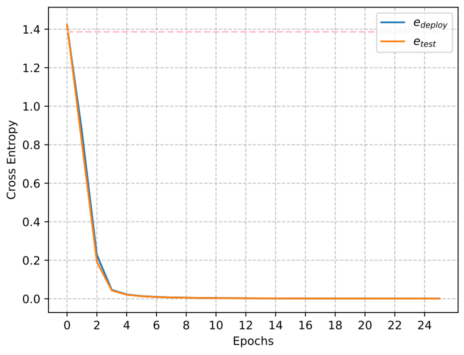
(a) $h_{\text{train}} = 2$



(b) $h_{\text{train}} = 4$



(c) $h_{\text{train}} = 6$



(d) $h_{\text{train}} = 8$

Figure 4.6.: Cross entropy deployment and test errors over increasing training epochs of the LSTM model in the GC4 scenario, computed for different training horizon $h_{\text{train}}$. The dashed line represents the loss of the baseline random classifier.

for $h_{\text{train}} = 12 > 10 = t_{\text{mix}}$ to tend to zero when the training horizon is around 22 time steps. More training time steps do not improve significantly the predictions. Thus, we select as training horizon $h^*_{\text{train}} = 22$. In the traffic grid domain, as illustrated in the bottom left panel of Figure 4.7, the deployment error of both learning models decreases substantially between 10 and 20 time steps. The model performance improves for training horizon ranging from 20 and 30 time steps. However, increasing further $h_{\text{train}}$ does not lead to performance improvement. This is confirmed by the training curves collected in Figure C.2 in Appendix C. Although the mixing time of the traffic grid is unknown, these results suggest that the system reaches a stationary influence between 10 and 30 time steps and $t_{\text{mix}}$ should be chosen within those values.

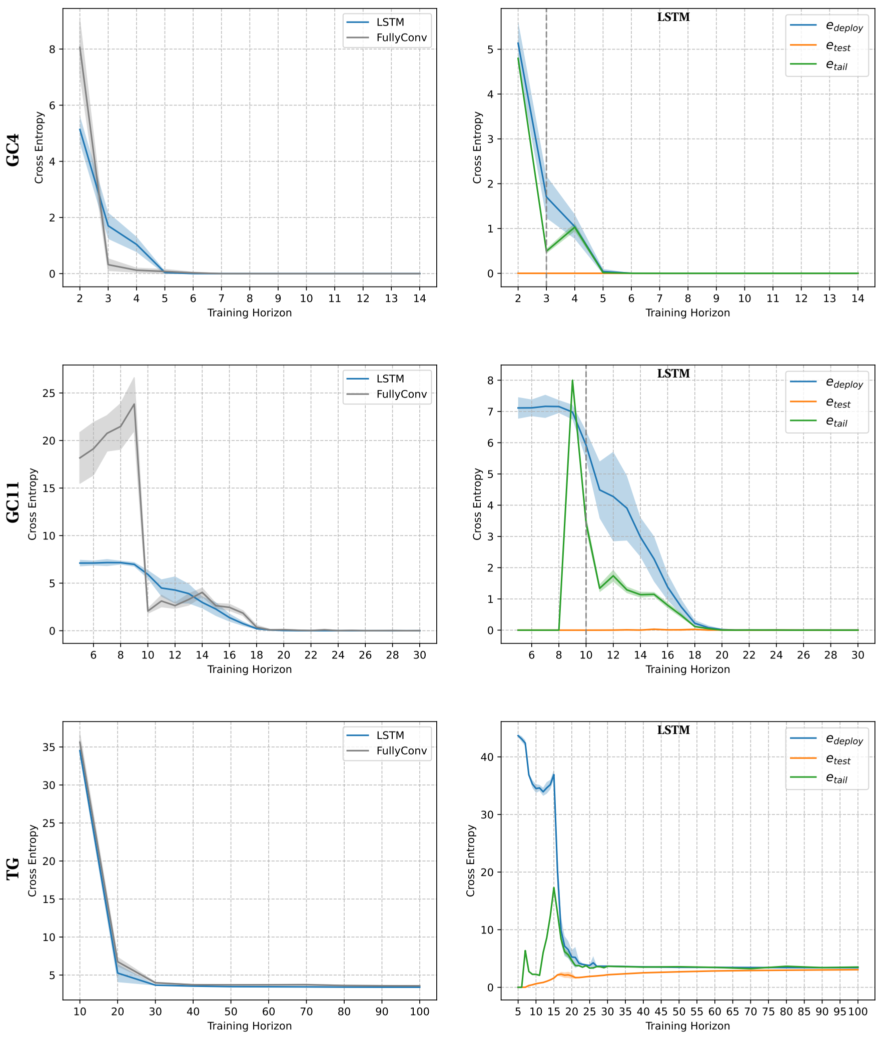


Figure 4.7.: Left side panels: deployment error over training horizons for the LTSM and FullyConv network. Right side panels: deployment, test and tail-test error for increasing training horizon for the LTSM model. The vertical dashed line marks the mixing time of the system, if known. Note that the apparent difference between the blue curves representing the LSTM error in the top and bottom plots is a result of the different scales of the other variables plotted or of different values of the training horizon considered.

**Deployment error estimate.** The right side panels of Figure 4.7 depict the deployment, test and test-tail errors of the LSTM model for different training horizons. The results from all three scenarios confirm uniformly the theoretical intuition presented in Section 4.3.3. When $h_{\text{train}} \geq t_{\text{mix}}$, the test-tail error (green) reflects the deployment error (blue) trend much better than the test error (orange). Specifically in the two grab a chair scenarios, we observe that after the mixing time, marked by the dashed vertical line, the test-tail error presents the same descending trend of the deployment error. Contrarily, the test error is always very close to zero, indicating that the model learns very well the influence over the training horizon. For the traffic domain, Figure 4.7 suggests that after approximately 15 time steps the system reaches the equilibrium. Thus, the deployment error and the test-tail error become very close and decrease quickly to converge to the test error. This analysis shows that the test-tail error provides a good estimate of the deployment error and can therefore be used to assess the performance of the learning model using the training set and to choose the optimal training horizon $h^*_{\text{train}}$ as the smallest horizon that yields a low test-tail error.

### 4.4.3. Observations, limitations and future work

Our study leads to a number of key observations, but there are also limits on how far the conclusions can reach. Here, we provide a balanced overview of both.

**Key observations.** The results of our empirical study show that, even in many complex real-life situations, the task of learning influence can often be quite simple and computationally manageable. This was especially evident in the case of microgrids, where a logistic regression model performs just as well as more advanced models. The complexity of the influence learning problem is affected by the dimensionality and horizon of the forecasting task. For high dimensional or long horizon problems, linear models or vanilla temporal convolutional networks typically fail to provide accurate influence approximations. However, relatively small recurrent and fully convolutional networks have proven to learn good approximations with efficient training times in all domains and thus be the most suitable models for the influence learning task. Additionally, the experiments show that suitable learning models can generalize the influence from a short horizon training trajectories well beyond the training horizon. In general, a good choice for such training horizon depends mostly on the mixing time of the system and it is independent of the learning model. In essence, an optimal training horizon corresponds to the mixing time plus a few additional time steps to collect enough experience of the stationary influence in the training set. The test-tail provides a better estimate of the deployment error and therefore can be used to assess the quality of the learning models and select an optimal horizon.

**Environment Scalability.** First, while this study has used relatively complex benchmark problems (microgrid with 100s of agents, and traffic control with more than 100 state variables), we did not exhaustively investigate the scalability of the environments. Scalability of influence prediction is mostly dependent on the number of influence sources that need to be predicted. As long as this is limited, the learning and use of influence can scale very well, as for instance demonstrated by Suau *et al.* [17] that employ influence-based abstraction, in parallel, to a traffic control task with 100 intersections. Furthermore, even though the IBA framework can be naturally extended to continuous state variables, our study is limited to discrete state variables, and does not address the more realistic challenge of learning continuous influence sources. Lastly, although the chosen domains are inspired by realistic scenarios, they still represent simplifications of actual real-world cases.

**Specification of local model and d-sets.** Additionally, our approach relies on the manual specification of local states and d-sets, focusing on the feasibility of influence prediction under these conditions. As discussed in Section 4.2.5, there are trade-offs between the predictability of influence sources and the size of the local model. Future research could explore automated methods for identifying good decomposition choices to optimize the overall performance.

**Transformers and other learning models.** We note that we have not exhaustively compared all possible sequence models. In particular, transformers, (discrete) diffusion models, and structured state space models could be of interest also to scale to higher dimensional influence prediction tasks. While such recent methods have pushed the limits in the large data regime, IBA is motivated by the idea that performing many simulations with a global simulator could be too expensive. Therefore our focus has been on exploring the potential of learning from limited amounts of data ($N = 500$ trajectories). Our study shows that exploring whether, for instance, (small) transformers could be competitive in this regime, or whether they could provide further benefits when problems become even more complex, is promising. However, these explorations are left for future research.

**Impact for control.** In this paper, we explored the feasibility of learning accurate influence predictors, as measured by CE loss. However, the full IBA framework intends to support decision making: in the end we care about the impact of using the resulting models in, e.g., planning, predictive control or RL. Intuitively the more accurate the influence predictors and hence I-ALM, the better results we would expect. Indeed, in prior work Congeduti, Mey, and Oliehoek [23] did investigate this question, showing theoretically that the CE error leads to an upper bound on value loss and empirically that there is a positive correlation between CE and value loss. Nevertheless, the relation between CE error and value loss is complex, and this alignment may not always hold [56, 57]. Specifically, using additional information about rewards or values can lead to tighter upper bounds and thus faster learning [56], which can have a significant impact in finite sample regimes [57]. Therefore, investigating these potential mismatches and adapting influence models to dynamic, interactive environments remains an important area for future exploration.

## 4.5. Conclusions

In this paper we investigated learning models and techniques for the influence learning task in realistic scenarios. We run an extensive empirical investigation of the performance of different learning models in a variety of domains. We conclude that complex scenarios may still induce manageable influence learning task. We showed that relatively small recurrent models can achieve the same performance levels as state-of-the-art fully convolutional neural networks. Moreover, we explored how to leverage learning models to build local simulators for long horizons using short training trajectories. In particular, we showed that there exists a training horizon which is sufficient to learn good influence approximations for long (or infinite) horizon problems and how to use suitable error metrics to search for such horizon. Approximate IBA offers a promising direction for managing real-world decision-making problems. However, inaccuracies of influence approximators in high-stakes domains, such as traffic management, energy distribution, and critical infrastructure may lead to unintended consequences including safety risks, economic inefficiencies or biases in decision-making. While this also holds when applying planning and RL techniques without IBA, IBA could potentially introduce further approximation errors. We recommend that future researchers, developers, and practitioners exploring IBA in applications prioritize rigorous validation to address these ethical concerns.

# REFERENCES

[1] E. Congeduti, R. Rocchetta, and F. A. Oliehoek. "Influence Learning in Complex Systems". In: *Transactions on Machine Learning Research (TMLR)* (2025).

[2] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems: concepts and design*. Addison-Wesley, 2001.

[3] P. Järventausta, S. Repo, A. Rautiainen, and J. Partanen. "Smart grid power system control in distributed generation environment". In: *Annual Reviews in Control* (2010).

[4] K. Nweye, S. Sankaranarayanan, and Z. Nagy. "MERLIN: multi-agent offline and transfer learning for occupant-centric operation of grid-interactive communities". In: *Applied Energy* (2023), p. 121323.

[5] G. Dimitrakopoulos and P. Demestichas. "Intelligent transportation systems". In: *IEEE Vehicular Technology Magazine* (2010), pp. 77–84.

[6] L. Kaelbling, M. Littman, and A. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* (1998), pp. 99–134.

[7] R. Sutton and A. Barto. *Reinforcement learning: an introduction*. MIT press, 2018.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, and G. Ostrovski. "Human-level control through deep reinforcement learning". In: *Nature* (2015), p. 529.

[9] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* (2016), pp. 484–489.

[10] S. M. Kakade. "On the sample complexity of reinforcement learning". PhD thesis. University College London (United Kingdom), 2003.

[11] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis. "Reinforcement learning, fast and slow". In: *Trends in Cognitive Sciences* (2019), pp. 408–422.

[12] F. Oliehoek, S. Witwicki, and L. Kaelbling. "Influence-based abstraction for multiagent systems". In: *AAAI Conference on Artificial Intelligence*. 2012.

[13] J. He, M. Suau de Castro, and F. Oliehoek. "Influence-augmented online planning for complex environments". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[14] J. He, M. Suau, H. Baier, M. Kaisers, and F. Oliehoek. “Online planning in POMDPs with self-improving simulators”. In: *International Joint Conference on Artificial Intelligence, (IJCAI)*. 2022.

[15] M. Suau, J. He, E. Congeduti, R. Starre, A. Czechowski, and F. Oliehoek. “Influence-aware memory architectures for deep reinforcement learning in POMDPs”. In: *Neural Computing and Applications* (2022), pp. 1–17.

[16] M. Suau, J. He, M. Spaan, and F. Oliehoek. “Influence-augmented local simulators: a scalable solution for fast deep RL in large networked systems”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2022.

[17] M. Suau, J. He, M. M. Çelikok, M. Spaan, and F. Oliehoek. “Distributed influence-augmented local simulators for parallel MARL in large networked systems”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022).

[18] Y. Shoham and K. Leyton-Brown. *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[19] F. Oliehoek, S. Witwicki, and L. Kaelbling. “A sufficient statistic for influence in structured multiagent environments”. In: *Journal of Artificial Intelligence Research (JAIR)* (2021), pp. 789–870.

[20] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[21] J. Gamboa. “Deep learning for time-series analysis”. In: *arXiv preprint arXiv:1701.01887* (2017).

[22] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. “Deep learning for time series classification: a review”. In: *Data Mining and Knowledge Discovery* (2019), pp. 917–963.

[23] E. Congeduti, A. Mey, and F. Oliehoek. “Loss bounds for approximate influence-based abstraction”. In: *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2021.

[24] R. Pascanu, T. Mikolov, and Y. Bengio. “On the difficulty of training recurrent neural networks”. In: *International Conference on Machine Learning (ICML)*. 2013.

[25] A. Gu and T. Dao. “Mamba: linear-time sequence modeling with selective state spaces”. In: *First Conference on Language Modeling*. 2024.

[26] E. Hansen and Z. Feng. “Dynamic programming for POMDPs using a factored state representation”. In: *International Conference on Artificial Intelligence Planning Systems (AIPS)*. 2000.

[27] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. “Taming decentralized POMDPs: towards efficient policy computation for multiagent settings”. In: *International Joint Conference on Artificial Intelligence, (IJCAI)*. 2003.

[28] E. Hansen, D. Bernstein, and S. Zilberstein. “Dynamic programming for partially observable stochastic games”. In: *AAAI Conference on Artificial Intelligence*. 2004.

[29] F. Oliehoek and C. Amato. “Best response Bayesian reinforcement learning for multiagent systems with state uncertainty”. In: *AAMAS Workshop on Multiagent Sequential Decision Making Under Uncertainty*. 2014.

[30] C. Boutilier, T. Dean, and S. Hanks. “Decision-theoretic planning: structural assumptions and computational leverage”. In: *Journal of Artificial Intelligence Research (JAIR)* (1999), pp. 1–94.

[31] D. Koller. *Probabilistic graphical models: principles and techniques*. 2009.

[32] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[33] S. Bai, Z. Kolter, and V. Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv preprint arXiv:1803.01271* (2018).

[34] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy. “Deep reinforcement learning for sequence-to-sequence models”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2019), pp. 2469–2489.

[35] J. Vazquez-Canteli, S. Dey, G. Henze, and Z. Nagy. “CityLearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management”. In: *arXiv preprint arXiv:2012.10504* (2020).

[36] P. Poupart and C. Boutilier. “VDCBPI: an approximate scalable algorithm for large POMDPs”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2004.

[37] R. Rocchetta, A. Mey, and F. Oliehoek. “A survey on scenario theory, complexity, and compression-based learning and generalization”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[38] F. Karim, S. Majumdar, H. Darabi, and S. Chen. “LSTM fully convolutional networks for time series classification”. In: *IEEE Access* (2017), pp. 1662–1669.

[39] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* (1997), pp. 1735–1780.

[40] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning phrase representations using RNN encoder–decoder for statistical machine translation”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[41] C. Lea, M. Flynn, R. Vidal, A. Reiter, and G. Hager. “Temporal convolutional networks for action segmentation and detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[42] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[44] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. “Transformers in time series: A survey”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2023.

[45] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: a review and new perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013), pp. 1798–1828.

[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *IEEE* (1998), pp. 2278–2324.

[47] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[48] A. Simeonov, Y. Du, B. Kim, F. Hogan, J. Tenenbaum, P. Agrawal, and A. Rodriguez. “A long horizon planning framework for manipulating rigid pointcloud objects”. In: *Conference on Robot Learning*. 2020.

[49] K. Pertsch, O. Rybkin, F. Ebert, S. Zhou, D. Jayaraman, C. Finn, and S. Levine. “Long-horizon visual planning with goal-conditioned hierarchical predictors”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[50] C. Dann and E. Brunskill. “Sample complexity of episodic fixed-horizon reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.

[51] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. Springenberg. “Learning by playing solving sparse reward tasks from scratch”. In: *International Conference on Machine Learning (ICML)*. 2018.

[52] M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[53] T. Morton and W. Wecker. “Discounting, ergodicity and convergence for Markov decision processes”. In: *Management Science* (1977).

[54] M. Kearns and S. Singh. “Near-optimal reinforcement learning in polynomial time”. In: *Machine learning* (2002).

[55] K. Miettinen. *Nonlinear multiobjective optimization*. Springer Science & Business Media, 1999.

[56] A.-m. Farahmand, A. Barreto, and D. Nikovski. “Value-aware loss function for model-based reinforcement learning”. In: *International Conference on Artificial Intelligence and Statistics*. 2017.

[57] N. Lambert, B. Amos, O. Yadan, and R. Calandra. “Objective mismatch in model-based reinforcement learning”. In: *Conference on Learning for Dynamics and Control*. 2020.

# 5 Conclusion

This dissertation provides a systematic characterization of state abstraction approaches across research fields, identifying underlying relationships, and deriving a general framework. Building on this foundation, it then focuses on the problem of modeling and learning external influences for local state abstractions in complex systems. Theoretical guarantees are established that relate the quality of influence approximations to the performance of the decision maker. An empirical investigation further demonstrates that good approximations of the influence can be learned efficiently, even in large-scale and long-horizon environments. These findings underscore the potential of influence-based abstraction (IBA) as a principled approach to scalable and effective decision making in complex environments.

This work contributes to a broader understanding of how intelligent systems can cope with complexity through principled simplification. By demonstrating that local abstractions can be efficiently learned, it shows how complex decision-making problems can be reformulated in more tractable terms. A central insight is that the influence of the global context on local decision problems is frequently not arbitrarily complex but exhibits structure that makes it both compactly representable and learnable. This highlights that local reasoning, when supported by well-chosen representations of external context, can serve as a powerful strategy for navigating complexity.

## 5.1. Key contributions and discussion

Here we discuss the main contributions.

### 5.1.1. A characterization for state abstraction

Chapter 2 synthesizes various state abstraction methods into a unified framework, bridging approaches from reinforcement learning, planning, robust control, and game theory. By examining the relationships among different abstraction schemes, we show how some approaches can be viewed as specific instances or refinements of others. This analysis gives rise to a chain of inclusion among the approaches examined, clarifying both their relative generality and the characteristics that distinguish them.

A central insight is that what fundamentally differentiates these abstraction schemes is how they account for the uncertainty introduced by state aggregation. Specifically, the uncertainty corresponds to the non-Markovian effects that occur when transitions in the abstract space no longer preserve the Markov property. Building on this insight, the thesis introduces a general framework for state abstraction based on how such uncertainty is modeled. The general definition encompasses all the approaches examined, each resulting from the adoption of specific structural assumptions. This unifying perspective supports a systematic understanding of abstraction methods and provides a foundation for the development, adaptation, and reuse of abstraction techniques across domains.

Additionally, in Appendix A we provide a formal proof of the bounds on the value loss incurred when solving a decision-making problem using an abstract model. Performance guarantees for abstract Markov decision processes have been widely studied, with several works providing bounds on the value loss [1–5]. Nonetheless, the existing literature presents some limitations: certain results establish an upper bound that grows cubically with the discount factor, while some proofs rely on an incorrect application of the triangle inequality, and most results are restricted to the infinite horizon setting. To the best of our knowledge, our work provides the first formally correct proof that includes both finite and infinite horizons and establishes a quadratic upper bound in terms of the discount factor (or problem horizon). In particular, in Theorem 5 the upper bound of the value loss is a function of several factors: some related to the choice of the abstraction model such as the reward error $\varepsilon_R$, the transition error $\varepsilon_T$, and the number of abstract states $|\bar{S}|$; others reflecting structural properties of the system, such as the planning horizon or the discount factor. The bound can be schematically formulated as

$$L(\varepsilon_R, \varepsilon_T) \leq C_1\,\varepsilon_R + C_2 \varepsilon_T |\bar{S}| \tag{5.1}$$

The loss $L$ refers to the difference between the optimal value in the ground state space and the value achieved by the optimal abstract policy, expressed as a function of the transition and reward errors. The bound shows that this loss is upper bounded by a linear function of the approximation errors $\varepsilon_R, \varepsilon_T$. The constants $C_1, C_2$ depend on fixed features of the environment such as the planning horizon or the discount factor.

A constructive example is provided to demonstrate that the bound is tight in the worst case. This result guarantees that, in the worst case, the performance loss increases linearly with the transition and reward errors as well as with the number of abstract states.

Intuitively, this result suggests a strategy for constructing abstraction models: one should aim to minimize $\varepsilon_R$, $\varepsilon_T$, or $|\bar{S}|$ to control the performance loss. However, these parameters are inherently interdependent: reducing reward or transition errors typically requires distinguishing more ground states, thereby increasing the number of abstract states; conversely, using fewer abstract states tends to increase the mismatch across aggregated states. Although the bound is tight in the worst case, there is no systematic study of the gap between the upper bound and the actual value loss in general settings, nor a clear understanding of the trade-off between abstraction accuracy and the number of abstract states. McCallum [6] observed that a general state abstraction scheme may in fact prevent the agent from finding an optimal policy. This limitation is further illustrated by the example in Section A.4, where a specific aggregation map results in the highest value loss while still exactly matching the upper bound. Although the bound is tight and perfectly informative in this case, the aggregation map leads to the poorest performance.

As a result, the bound, while theoretically sound, offers limited practical guidance for abstraction design in general settings unless additional structural assumptions are made. In light of these considerations, this thesis narrows its focus to localized abstractions that leverage structural properties of environments, such as factorization.

### 5.1.2. PERFORMANCE GUARANTEES FOR INFLUENCE APPROXIMATIONS

Chapter 3 establishes theoretical performance guarantees for approximate IBA and complements them with aligned empirical observations.

Central contributions are analytical and probabilistic formulations of upper bounds on the value loss for using approximations of the influence. Specifically, Corollary 1 and Theorem 4 show that the value loss can be bounded by a function of the approximation error. A compact reformulation of these results, intended as a qualitative description to highlight the relationship between the value loss and the approximation error, can be expressed as

$$L(\varepsilon_I) \leq C\varepsilon_I \tag{5.2}$$

where $\varepsilon_I$ denotes the influence approximation error, defined by the square root of the Kullback-Leibler (KL) divergence between the true and approximate influence distributions. The constant $C$ summarizes structural factors of the environment, such as the planning horizon or discount factor.

These bounds offer a strong theoretical link between the problem of minimizing the value loss and the principle of maximum likelihood estimation for learning influence approximations. Since the KL divergence corresponds to the expected negative log-likelihood (up to an additive constant), minimizing the influence approximation error aligns with likelihood maximization. This argument provides a principled justification for using deep learning models trained to minimize cross-entropy loss.

The bound expressed in Equation (5.2) shares the same structure as the general state abstraction bound in Equation (5.1). In fact, both approximate influence models and state abstractions can be understood as approximations of the underlying ground-truth model. Specifically, an approximate influence model introduces only an error in the transition dynamics, preserving the exact reward structure. For this reason, the reward error term does not appear in Equation (5.2), while the influence approximation error $\varepsilon_I$ captures the deviation in transition modeling, corresponding to the transition error term $\varepsilon_T$ in the state abstraction bound Equation (5.1). Moreover, the constants $C$ and $C_2$ reflect analogous environmental dependencies: both are quadratic in the planning horizon and linear in the maximum absolute reward.

However, there are fundamental differences between the two bounds. First, the bound for influence approximations does not depend on the size of the state space. Moreover, the influence approximation error $\varepsilon_I$ is not tied to structural properties of the system, such as similarities between states, but depends solely on the quality of the approximations. As a result, assuming that accurate influence approximations can be learned, the error $\varepsilon_I$ can be made arbitrarily small. Therefore, under the general assumption that accurate influence approximations can be learned, the value loss tends to zero at least linearly with the approximation error $\varepsilon_I$, i.e. $L(\varepsilon_I) = O(\varepsilon_I)$. The fact that the value loss becomes arbitrarily small by improving the accuracy of the influence approximation, regardless of the environment characteristics and complexity, makes this approach potentially better suited for large-scale environments.

Finally, the reported empirical results support the theoretical findings by showing in simple scenarios that the cross-entropy loss correlates with the performance loss.

### 5.1.3. Empirical evidence for the learnability of influence

Motivated by the discussion on the theoretical guarantees established, Chapter 4 investigates the practical feasibility of learning accurate influence approximations in complex environments. Through a series of experiments across several realistic domains, it demonstrates that influence representations can be learned reliably and efficiently. A key finding is that relatively small recurrent neural networks are sufficient to model the influence even in complex, high-dimensional scenarios. Under ergodic assumptions, these learning models also generalize effectively to prediction horizons significantly longer than those used during training. In addition, the chapter proposes a simple and effective method to estimate the prediction error over longer horizons, helping to ensure reliable performance during deployment.

This indicates that influence learning is not only tractable but also scalable to large and temporally extended decision-making scenarios. Together, these results support the tractability of IBA as a powerful abstraction method for complex decision-making systems.

## 5.2. Real-world challenges

Learning local representations in real-world domains introduces several practical challenges. In Catalán Pastor, Congeduti, and Oliehoek [7], we presented a preliminary investigation into the problem of learning local models from historical data, in the context of a real-world traffic application. The work specifically addresses the problem of missing data of traffic volumes caused by sensor malfunctions, using traffic forecasting models to reconstruct the values at faulty sensor locations from information provided by neighboring sensors. Although the focus is on this specific task, the study reveals several limitations that may arise when learning local abstractions from real-world data in sequential decision-making scenarios modeled as factored Markov decision processes [8].

The following observations that are consistent with the limitations identified in traffic forecasting literature [9, 10]. Although grounded in the specific context of traffic forecasting, these limitations represent broader challenges that should be considered when learning local models from real-world data in factored Markov models.

An important aspect to consider is the violation of the Markov assumption. In a first-order Markov model, the next state of the system is assumed to depend only on its current state, with no memory of how that state was reached. In practice, this implies that there is no memory of how the environment reached its current state. However, in real-world traffic scenarios, vehicle behavior, and consequently the observed traffic volumes, is often influenced by longer-term dependencies. For example, the likelihood of a vehicle turning at an intersection may depend not only on its current position but also on its prior trajectory: vehicles arriving from highways may be more likely to continue along arterial roads, while those coming from local streets may head into nearby neighborhoods. Such routing preferences reflect the dependency on the path history. Similarly, long-term congestion effects play a role: drivers often adjust their routes based on earlier traffic conditions or anticipated peak-hour patterns. As a result, current decisions may be shaped by past congestion states that are no longer directly observable. These temporal dependencies break the Markov assumption and cannot be captured by a first-order Markov model.

A second challenge is the non-stationarity of transitions. Traffic patterns vary significantly over time due to factors such as rush hours, weekends, and seasonal effects, making the transition dynamics inherently time dependent. Consequently, a fixed, time-invariant transition function may not well represent traffic dynamics. External variables may also play a significant role, with factors such as traffic light phases, road conditions, traffic regulations, and weather conditions strongly influencing traffic flows. These hidden effects introduce uncertainty that is difficult to capture and make it more challenging to learn reliable local representations.

An important consideration when analyzing these aspects is the time scale at which traffic is modeled. Over short time intervals, the system is less affected by historical dependencies or external effects. At a finer temporal resolution, the stationarity and Markov assumptions are generally more reasonable approximations: a vehicle's immediate next move is often strongly correlated with its current position, speed, and local context while external factors such as weather or road conditions typically remain unchanged. However, even when such assumptions are theoretically justified, practical constraints, such as sensor sampling rates, often limit the temporal resolution of available data, making it difficult to capture the dynamics with appropriate granularity.

Finally, data quality is a critical factor. For example, measurements from real-world loop detector sensors are often noisy, sparse, or incomplete. Missing values and irregular sampling can hinder the estimation of local transition probabilities and degrade the reliability of learned local influences.

## 5.3. Limitations and future work

We outline the main limitations of the current work and identify promising directions for future research, which are discussed in the subsections below.

### 5.3.1. Influence search in multiagent systems

In this thesis, we adopt the perspective of a single agent learning the local influence induced by potentially many other agents with fixed policies, in order to compute a best-response policy. This perspective is particularly relevant in multiagent systems, as best-response computation plays a central role in many algorithms for finding Nash equilibria [11, 12]. Nevertheless, further investigation is needed to fully understand the potential of approximate IBA in joint policy optimization.

A key insight is that the influence experienced by a local agent captures only those aspects of other agent's policies that are relevant to its own decision-making problem. As a result, different external policies may induce the same local influence. This implies that the space of joint influences can be significantly smaller than the space of joint policies. This suggests a promising direction: searching for equilibria in the influence space rather than in the policy space [13]. This direction raises several open questions regarding the formulation and solution of optimization problems in the space of joint influences. First, it is necessary to define an appropriate notion of equilibrium in the influence space and to identify the conditions under which such an equilibrium corresponds to a valid solution in the original policy space. Furthermore, the definition of the influence space itself requires careful consideration: influences are, in principle, probability distributions over certain factors of the state space, and thus belong to subsets of an infinite-dimensional space. Finally, even if an optimal joint influence is identified, recovering a corresponding joint policy may require solving constrained optimization problems for each agent to ensure that their policies induce the desired joint influence. This reconstruction process can be computationally demanding and may not always admit a feasible solution.

In future work, our aim is to develop a formal framework for influence-based equilibrium analysis and search in multiagent systems. This includes characterizing the structure of the influence space, identifying conditions under which influence equilibria correspond to equilibria in the policy space, and analyzing the convergence properties of iterative solution methods.

### 5.3.2. Impact of influence approximations on the control problem

Although this thesis offers theoretical guarantees and empirical evidence that minimizing the cross entropy loss for influence approximations is well aligned with the control goal of minimizing the value loss, the analysis remains limited in certain aspects.

The relationship between cross entropy and value loss is inherently complex. One example consists of systems with highly decoupled local dynamics. In such cases, the influence strength [14], defined as the extent to which influence affects the local dynamics or the local value, may be limited. Therefore, a better quality of the approximations does not directly imply better performance. Moreover, misalignment can arise when structural assumptions about the reward function or value hold [15, 16]. In such cases, more informed loss functions may be better suited to achieving lower value loss. A more precise characterization of these scenarios could guide approximation strategies and help define target accuracies.

Additionally, the bounds presented in this work are based on a maximization over all possible d-separating sets, which reflects only the worst-case cross entropy error. While this provides a general guarantee, it may result in overly loose upper bounds on the value loss in typical scenarios. A formulation based on expected error could offer a more informative perspective on average-case behavior. Furthermore, additional assumptions about the structure of rewards or value functions could also be leveraged to derive tighter bounds that more accurately reflect the relationship between approximation quality and control performance.

### 5.3.3. LOCAL DECOMPOSITION CHOICES

Our work assumes a fixed decomposition into local and external factors. Although our experiments suggest that suitable decompositions are often easy to identify, different choices can lead to influence learning problems of varying complexity. In particular, larger local models include more information, which can make the influence prediction task easier but at the cost of reduced computational efficiency during simulation and training. Moreover, the choice of the decomposition also affects the strength of the external influence on the local dynamics. Expanding the local model may introduce variables that are more tightly coupled with the external environment, thus increasing the dependence on accurate influence predictions. As such, the local control task may become more sensitive to errors in the learned influence. In essence, selecting a local decomposition requires reasoning about computational efficiency, tractability of the influence prediction task, and the strength of external influence on local control.

Future work should systematically investigate this trade-off, potentially identifying scenarios where the structure or size of the local model has a greater impact on the overall performance in local control. Moreover, we aim to explore whether local decompositions can be learned automatically based on all these considerations. If we can overcome the current limitations related to the decomposition choice, we expect that our techniques could make an impact on decision making in complex environments.

# REFERENCES

[1] R. Dearden and C. Boutilier. “Abstraction and approximate decision-theoretic planning”. In: *Artificial Intelligence* (1997), pp. 219–283.

[2] L. Li, T. J. Walsh, and M. L. Littman. “Towards a unified theory of state abstraction for MDPs”. In: *International Symposium on Artificial Intelligence and Mathematics (AI&Math)*. 2006.

[3] B. Van Roy. “Performance loss bounds for approximate value iteration with state aggregation”. In: *Mathematics of Operations Research* (2006), pp. 234–244.

[4] D. Abel, D. Hershkowitz, and M. Littman. “Near optimal behavior via approximate state abstraction”. In: *International Conference on Machine Learning (ICML)*. 2016.

[5] D. Abel. “A theory of state abstraction for reinforcement learning”. In: *AAAI Conference on Artificial Intelligence*. 2019.

[6] A. K. McCallum. *Reinforcement learning with selective perception and hidden state*. University of Rochester, 1997.

[7] V. Catalán Pastor, E. Congeduti, and F. Oliehoek. “Overcoming traffic sensors malfunctions with deep learning”. In: *Benelux Conference on Artificial Intelligence (BNAIC) and Belgian Dutch Conference on Machine Learning (BeNeLearn)*. 2022.

[8] C. Boutilier, R. Dearden, and M. Goldszmidt. “Stochastic dynamic programming with factored representations”. In: *Artificial Intelligence* (2000), pp. 49–107.

[9] B. Yu, H. Yin, and Z. Zhu. “Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2018.

[10] J. Zhang, Y. Zheng, and D. Qi. “Deep spatio-temporal residual networks for citywide crowd flows prediction”. In: *AAAI conference on artificial intelligence*. 2017.

[11] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel. “A unified game-theoretic approach to multiagent reinforcement learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[12] B. McMahan, G. Gordon, and A. Blum. “Planning in the presence of cost functions controlled by an adversary”. In: *International Conference on Machine Learning (ICML)*. 2003.

[13] S. Witwicki and E. Durfee. “Influence-based policy abstraction for weakly-coupled Dec-POMDPs”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2010.

[14] J. He, M. Suau de Castro, and F. Oliehoek. “Influence-augmented online planning for complex environments”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[15] A.-m. Farahmand, A. Barreto, and D. Nikovski. “Value-aware loss function for model-based reinforcement learning”. In: *International Conference on Artificial Intelligence and Statistics*. 2017.

[16] A. Ayoub, Z. Jia, C. Szepesvari, M. Wang, and L. Yang. “Model-based reinforcement learning with value-targeted regression”. In: *International Conference on Machine Learning (ICML)*. 2020.

# A

# On value bounds for state abstraction

## A.1. Introduction and notation

The problem of establishing performance guarantees when optimizing a Markov decision process (MDP) in an abstract state space has been extensively studied, with various works providing value bounds [1–4]. However, to the best of our knowledge, this work provides the first correct formal proof for a general state abstraction framework covering both finite and infinite horizon settings. The main result, inspired by the work [1], establishes upper bounds on the value loss incurred when aggregating states that differ in their transition and reward functions under an approximate abstraction scheme. The derived bounds also show that, in the exact abstraction case, no value loss occurs. Additionally, we demonstrate the tightness of the bound in the worst-case sense by constructing an example of state abstraction for which the value loss matches the upper bound.

We consider an MDP, $\mathcal{M} = (S, A, T, R)$ with state space $S$, action space $A$, transition function $T$ and reward function $R$. We discuss separately the infinite horizon case with discount factor $\gamma < 1$ and the finite horizon case with horizon $h$. We assume that the first reward is collected at time $t = 0$. This implies that, for a horizon $h$, the agent will collect $h+1$ rewards. We consider an aggregation function that maps the ground tr ue space $S$ into the abstract space $\phi_{\bar{S}} : S \longrightarrow \bar{S}$. We denote an abstract MDP as $\bar{\mathcal{M}} = (\bar{S}, A, \bar{T}, \bar{R})$, for an arbitrary choice of the abstract transition and reward functions $\bar{T}$ and $\bar{R}$.

First, we introduce the notation.

- $\bar{\pi}$ is an abstract policy defined over the abstract state space $\bar{S}$. With a slight abuse of notation, $\bar{\pi}(s)$ denotes the extension of the abstract policy to the ground state space as $\bar{\pi}(s) \triangleq \bar{\pi}(\phi_{\bar{S}}(s))$.

- The expression $s \in \bar{s}$ indicates that the ground state $s$ corresponds to the abstract state $\bar{s}$. More precisely, $s$ belongs to the preimage of $\bar{s}$ under the aggregation map: $s \in \phi_{\bar{S}}^{-1}(\bar{s})$.

- $\max|R| \triangleq \max_{a,s}|R(s,a)|$ denotes the maximum absolute value of the reward function.
- $|\bar{S}|$ denotes the number of states in the abstract state space.
- $\bar{V}^{\bar{\pi}}$, $V^{\pi}$ corresponds to the value for the policy $\bar{\pi}$ and $\pi$ respectively in the abstract and ground spaces.
- $\bar{V}^*$ and $V^*$, $\bar{\pi}^*$ and $\pi^*$ are the optimal values and policies in the abstract and ground space, respectively.

## A.2. Preliminaries

We define $\mathcal{T}^{\pi}$ as the Bellman operator for a policy $\pi$ defined over the space of the value functions $V : S \longrightarrow \mathbb{R}$ as

$$(\mathcal{T}^{\pi}V)(s) = \begin{cases} R(s,\pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) V(s') & \text{infinite horizon} \\ R(s,\pi(s)) + \sum_{s'} T(s' \mid s, \pi(s)) V(s') & \text{finite horizon} \end{cases}$$

Then, we introduce the sequence obtained by repeatedly applying the Bellman operator as

$$\begin{aligned} V^{\pi,t}(s) &= \left(\mathcal{T}^{\pi} V^{\pi,t-1}\right)(s) \quad \text{for any } t > 0 \\ V^{\pi,0}(s) &= R(s,\pi(s)) \end{aligned} \tag{A.1}$$

In the infinite horizon case, $\mathcal{T}^{\pi}$ is a contraction with fixed point $V^{\pi}$ [5]. Therefore, the sequence converges to the value of the policy, as $V^{\pi,t}(s) \to V^{\pi}(s)$. In the finite horizon case, at iteration $h$, the sequence corresponds to the value of the policy $\pi$, meaning $V^{\pi,h}(s) = V^{\pi}(s)$.

Likewise, we define the optimal Bellman operator $\mathcal{T}^* = \mathcal{T}^{\pi^*}$ as

$$(\mathcal{T}^* V)(s) = \begin{cases} \max_a \left( R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) V(s') \right) & \text{infinite horizon} \\ \max_a \left( R(s,a) + \sum_{s'} T(s' \mid s, a) V(s') \right) & \text{finite horizon} \end{cases}$$

and the sequence obtained by repeatedly applying the optimal Bellman operator as

$$\begin{aligned} V^{*,t}(s) &= \left(\mathcal{T}^* V^{*,t-1}\right)(s) \quad \text{for any } t > 0 \\ V^{*,0}(s) &= \max_a R(s,a) \end{aligned} \tag{A.2}$$

In the infinite horizon case, $\mathcal{T}^*$ is a contraction with fixed point $V^*$ [5]. Therefore, the sequence converges to the optimal value, as $V^{*,t}(s) \to V^*(s)$. In the finite horizon case, at iteration $h$, the sequence corresponds to the optimal value, meaning $V^{*,h}(s) = V^*(s)$.

We first show a property of this sequence that will be used to prove the main result.

**Proposition 2.** For every policy $\pi$.

$$\|V^{\pi,t}\|_\infty \leq \begin{cases} \dfrac{\max|R|}{1-\gamma}, & \forall t \geq 0 \quad \text{infinite horizon} \\ (t+1)\max|R|, \forall h \geq t \geq 0 & \text{finite horizon} \end{cases} \tag{A.3}$$

*Proof.* Let us start by considering the sequence in the infinite horizon case. The following inequality follows from applying the triangular inequality and the maximum over the states.

$$\begin{aligned} |V^{\pi,t}(s)| &= \left|R(s,\pi(s)) + \gamma \sum_{s'} T(s' \mid s,\pi(s)) V^{\pi,t-1}(s)\right| \\ &\leq \max|R| + \gamma \|V^{\pi,t-1}\|_\infty \sum_{s'} T(s' \mid s,\pi(s)) = \max|R| + \gamma \|V^{\pi,t-1}\|_\infty. \end{aligned}$$

If we take the maximum over the states on the left hand side of the inequality and iterate the inequality over decreasing values of $t$, we get

$$\|V^{\pi,t}\|_\infty \leq \max|R| + \gamma \|V^{\pi,t-1}\|_\infty \leq \cdots \leq \max|R| + \gamma \max|R| + \cdots + \gamma^t \|V^0\|_\infty.$$

Thus,

$$\|V^{\pi,t}\|_\infty \leq \max|R| \sum_{i=0}^{t} \gamma^i \leq \max|R| \sum_{t=0}^{\infty} \gamma^i = \frac{\max|R|}{1-\gamma}.$$

Retracing the same steps for the finite horizon case, we demonstrate that for every $t \leq h$

$$\|V^{\pi,t}\|_\infty \leq \max|R| \sum_{i=0}^{t} 1 = (t+1)\max|R|.$$

□

## A.3. A FORMAL PROOF

**Theorem 5.** Assume that for given constants $\varepsilon_T$ and $\varepsilon_R$, the abstract transition and reward functions satisfy

$$\left|\bar{T}(\bar{s}' \mid \bar{s}, a) - \sum_{s' \in \bar{s}'} T(s' \mid s, a)\right| \leq \varepsilon_T \tag{A.4}$$

$$\left|\bar{R}(\bar{s}, a) - R(s, a)\right| \leq \varepsilon_R \tag{A.5}$$

for every $\bar{s}, \bar{s}' \in \bar{S}$, $s \in \bar{s}$, $a \in A$. Then, the following bounds on the value loss hold

$$V^*(s) - V^{\bar{\pi}^*}(s) \leq \begin{cases} \dfrac{2\varepsilon_R}{(1-\gamma)} + \dfrac{2\varepsilon_T \gamma \max|R|\,|\bar{S}|}{(1-\gamma)^2} & \text{infinite horizon} \\ 2\varepsilon_R(h+1) + \varepsilon_T(h+1)h\max|R|\,|\bar{S}| & \text{finite horizon} \end{cases}$$

Before starting the proof, note that the assumption (A.4) expresses that the probability of transition from a ground state $s$ into the cluster of ground states represented by the abstract state $\bar{s}' \in \bar{S}$, i.e., $\mathbb{P}(\bar{s}' \mid s, a) = \sum_{s' \in \bar{s}'} T(s' \mid s, a)$, should not be too different from the probability of transition from the abstract state $\bar{s}$ to $\bar{s}'$ in the abstract space.

We split the proof into two lemmas. The first lemma relates the abstract value to the ground value of an abstract policy, while the second establishes the relationship between the optimal values in the abstract and ground MDPs.

**Lemma 2.** Under the assumptions (A.4), (A.5), for every abstract policy $\bar{\pi}$ and state $s \in \bar{s}$,

$$\left|\bar{V}^{\bar{\pi}}(\bar{s}) - V^{\bar{\pi}}(s)\right| \leq \begin{cases} \dfrac{\varepsilon_R}{1-\gamma} + \dfrac{\varepsilon_T \gamma \max|R|\,|\bar{S}|}{(1-\gamma)^2} & \text{infinite horizon} \\ \varepsilon_R (h+1) + \varepsilon_T \dfrac{(h+1)h}{2} \max|R|\,|\bar{S}| & \text{finite horizon} \end{cases}$$

*Proof.* We consider the sequences $\bar{V}^{\bar{\pi},t}$,and $V^{\bar{\pi},t}$ defined in (A.1) by the Bellman operators on the ground and abstract spaces, respectively. First, we show by induction that in the infinite horizon case

$$\left|\bar{V}^{\bar{\pi},t}(\bar{s}) - V^{\bar{\pi},t}(s)\right| \leq \varepsilon_R \gamma^t + \left(\varepsilon_R + \varepsilon_T \gamma \frac{\max|R|}{1-\gamma} |\bar{S}|\right) \sum_{i=0}^{t-1} \gamma^i, \tag{A.6}$$

for any $t \geq 0$, $\bar{s}$ and $s \in \bar{s}$.

**Base case.** By the assumption (A.5) for $t = 0$

$$\left|\bar{V}^{\bar{\pi},0}(\bar{s}) - V^{\bar{\pi},0}(s)\right| = |\bar{R}(\bar{s}, \bar{\pi}(\bar{s})) - R(s, \bar{\pi}(\bar{s}))| \leq \varepsilon_R.$$

**Inductive step.** By the triangular inequality and rearranging the terms in the sum, we obtain

$$\left|\bar{V}^{\bar{\pi},t}(\bar{s}) - V^{\bar{\pi},t}(s)\right| \leq \left|\bar{R}(\bar{s}, \bar{\pi}(\bar{s})) - R(s, \bar{\pi}(\bar{s}))\right| \\ + \gamma \left| \sum_{\bar{s}' \in \bar{S}} \bar{T}(\bar{s}' \mid \bar{s}, \bar{\pi}(\bar{s})) \bar{V}^{\bar{\pi},t-1}(\bar{s}') - \sum_{\bar{s}' \in \bar{S}} \sum_{s' \in \bar{s}'} T(s' \mid s, \bar{\pi}(\bar{s})) V^{\bar{\pi},t-1}(s') \right|.$$

Now we use the assumption (A.5), add and subtract the term $\sum_{\bar{s}' \in \bar{S}} \bar{V}^{\bar{\pi},t-1}(\bar{s}') \sum_{s' \in \bar{s}'} T(s' \mid s, \bar{\pi}(\bar{s}))$ to the absolute value on right-hand side of the previous inequality, and use the triangular inequality to derive

$$\begin{aligned}\left|\bar{V}^{\bar{\pi},t}(\bar{s})-V^{\bar{\pi},t}(s)\right| &\leq \varepsilon_R+\gamma\left|\sum_{\bar{s}'\in\bar{S}}\bar{T}(\bar{s}'\mid\bar{s},\bar{\pi}(\bar{s}))\bar{V}^{\bar{\pi},t-1}(\bar{s}')-\sum_{\bar{s}'\in\bar{S}}\bar{V}^{\bar{\pi},t-1}(\bar{s}')\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s}))\right|\\ &\quad+\gamma\left|\sum_{\bar{s}'\in\bar{S}}\bar{V}^{\bar{\pi},t-1}(\bar{s}')\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s}))-\sum_{\bar{s}'\in\bar{S}}\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s}))V^{\bar{\pi},t-1}(s')\right|\\ &\leq \varepsilon_R+\gamma\|\bar{V}^{\bar{\pi},t-1}\|_\infty\sum_{\bar{s}'}\left|\bar{T}(\bar{s}'\mid\bar{s},\bar{\pi}(\bar{s}))-\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s}))\right|\\ &\quad+\gamma\sum_{\bar{s}'}\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s}))\left|\bar{V}^{\bar{\pi},t-1}(\bar{s}')-V^{\bar{\pi},t-1}(s')\right|.\end{aligned}$$

Now using (A.3) on the first term and the inductive hypothesis (A.6) for $t-1$ on the second term of the sum, we obtain

$$\begin{aligned}\left|\bar{V}^{\bar{\pi},t}(\bar{s})-V^{\bar{\pi},t}(s)\right| &\leq \varepsilon_R+\gamma\frac{\max|R|}{1-\gamma}\sum_{\bar{s}'}\left|\bar{T}(\bar{s}'\mid\bar{s},\bar{\pi}(\bar{s}))-\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s}))\right|\\ &\quad+\gamma\left(\varepsilon_R\gamma^{t-1}+\left(\varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|\right)\sum_{i=0}^{t-2}\gamma^i\right)\sum_{\bar{s}'}\sum_{s'\in\bar{s}'}T(s'\mid s,\bar{\pi}(\bar{s})).\end{aligned}$$

We rearrange the terms and use the assumption (A.4) to conclude the inductive step as follows

$$\begin{aligned}\left|\bar{V}^{\bar{\pi},t}(\bar{s})-V^{\bar{\pi},t}(s)\right| &\leq \varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|+\gamma\left(\varepsilon_R\gamma^{t-1}+\left(\varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|\right)\sum_{i=0}^{t-2}\gamma^i\right)\\ &= \varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|+\varepsilon_R\gamma^t+\left(\varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|\right)\sum_{i=1}^{t-1}\gamma^i\\ &= \varepsilon_R\gamma^t+\left(\varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|\right)\sum_{i=0}^{t-1}\gamma^i.\end{aligned}$$

If we consider the limit for $t\to\infty$ of this inequality we obtain the claim as follows

$$\begin{aligned}\left|\bar{V}^{\bar{\pi}}(\bar{s})-V^{\bar{\pi}}(s)\right| &= \lim_{t\to\infty}\left|\bar{V}^{\bar{\pi},t}(\bar{s})-V^{\bar{\pi},t}(s)\right|\leq\lim_{t\to\infty}\varepsilon_R\gamma^t+\left(\varepsilon_R+\varepsilon_T\gamma\frac{\max|R|}{1-\gamma}|\bar{S}|\right)\sum_{i=0}^{t-1}\gamma^i\\ &= \frac{\varepsilon_R}{1-\gamma}+\frac{\varepsilon_T\gamma\max|R|\,|\bar{S}|}{(1-\gamma)^2}.\end{aligned}$$

For the finite horizon case, we prove by induction that for any $0\leq t\leq h$

$$\left|\bar{V}^{\bar{\pi},t}(\bar{s})-V^{\bar{\pi},t}(s)\right|\leq\varepsilon_R(t+1)+\varepsilon_T\frac{(t+1)t}{2}\max|R|\,|\bar{S}| \tag{A.7}$$

**Base case.** The case for $t=0$ follows directly from the infinite horizon base case.

**Inductive step.** Retracing the steps for the infinite horizon case yields

$$\begin{aligned}\left|\bar{V}^{\bar{\pi},t}(\bar{s}) - V^{\bar{\pi},t}(s)\right| \leq \varepsilon_R + \|\bar{V}^{\bar{\pi},t-1}\|_\infty \sum_{\bar{s}'} \left| \bar{T}(\bar{s}' \mid \bar{s}, \bar{\pi}(\bar{s})) - \sum_{s' \in \bar{s}'} T(s' \mid s, \bar{\pi}(\bar{s})) \right| \\ + \sum_{\bar{s}'} \sum_{s' \in \bar{s}'} T(s' \mid s, \bar{\pi}(\bar{s})) \left| \bar{V}^{\bar{\pi},t-1}(\bar{s}') - V^{\bar{\pi},t-1}(s') \right|.\end{aligned}$$

Now using (A.3) on the first term, the inductive hypothesis for $t-1$ on the second term, we conclude the inductive step as follows

$$\begin{aligned}\left|\bar{V}^{\bar{\pi},t}(\bar{s}) - V^{\bar{\pi},t}(s)\right| &\leq \varepsilon_R + \varepsilon_T\, t \max|R|\,|\bar{S}| + \varepsilon_R\, t + \varepsilon_T \frac{t(t-1)}{2} |R|\,|\bar{S}| \\ &= \varepsilon_R(t+1) + \varepsilon_T \frac{(t+1)t}{2} \max|R|\,|\bar{S}|.\end{aligned}$$

The expression (A.7) for $t = h$ corresponds to the claim. □

**Lemma 3.** Under the assumptions (A.4), (A.5), for every state $s \in \bar{s}$

$$\left|\bar{V}^*(\bar{s}) - V^*(s)\right| \leq \begin{cases} \dfrac{\varepsilon_R}{1-\gamma} + \dfrac{\varepsilon_T \gamma \max|R|\,|\bar{S}|}{(1-\gamma)^2} & \text{infinite horizon} \\ \varepsilon_R\,(h+1) + \varepsilon_T \dfrac{(h+1)h}{2} \max|R|\,|\bar{S}| & \text{finite horizon} \end{cases}$$

*Proof.* We prove that (A.6) and (A.7) hold for the sequence $V^{*,t}$ defined by the optimal Bellman operator in (A.2). Specifically, we replicate the same derivations of Lemma 2 and use the sublinearity of the maximum, i.e. $|\max f - \max g| \leq \max|f - g|$, to show by induction that for any $t \geq 0$

$$\left|\bar{V}^{*,t}(\bar{s}) - V^{*,t}(s)\right| \leq \begin{cases} \varepsilon_R \gamma^t + \left(\varepsilon_R + \varepsilon_T \gamma \dfrac{\max|R|}{1-\gamma} |\bar{S}|\right) \displaystyle\sum_{i=0}^{t-1} \gamma^i & \text{infinite horizon} \\ \varepsilon_R(t+1) + \varepsilon_T \dfrac{(t+1)t}{2} \max|R|\,|\bar{S}| & \text{finite horizon} \end{cases} \tag{A.8}$$

**Base case.** For both infinite and finite horizon, we have

$$\left|\bar{V}^{*,0}(\bar{s}) - V^{*,0}(s)\right| = \left|\max_a \bar{R}(\bar{s}, a) - \max_a R(s, a)\right| \leq \max_a |\bar{R}(\bar{s}, a) - R(s, a)| \leq \varepsilon_R.$$

**Inductive step.** We assume the statement holds for $t-1$ in the infinite horizon case, then

$$\begin{aligned}
\left|\bar{V}^{*,t}(\bar{s}) - V^{*,t}(s)\right| &\leq \max_a \Big|\bar{R}(\bar{s},a) - R(s,a) \\
&\qquad + \gamma \sum_{\bar{s}'} \Big( \bar{T}(\bar{s}' \mid \bar{s}, a) \bar{V}^{*,t-1}(\bar{s}') - \sum_{s' \in \bar{s}'} T(s' \mid s,a) V^{*,t-1}(s') \Big) \Big| \\
&\leq \max_a |\bar{R}(\bar{s},a) - R(s,a)| \\
&\qquad + \gamma \max_a \sum_{\bar{s}'} \Big| \bar{T}(\bar{s}' \mid \bar{s}, a) \bar{V}^{*,t-1}(\bar{s}') - \sum_{s' \in \bar{s}'} T(s' \mid s,a) V^{*,t-1}(s') \Big| \\
&\leq \varepsilon_R + \gamma \max_a \sum_{\bar{s}'} \Big| \bar{V}^{*,t-1}(\bar{s}') \Big( \bar{T}(\bar{s}' \mid \bar{s}, a) - \sum_{s' \in \bar{s}'} T(s' \mid s,a) \Big) \Big| \\
&\qquad + \gamma \max_a \sum_{\bar{s}'} \sum_{s' \in \bar{s}'} T(s' \mid s,a) \left| \bar{V}^{*,t-1}(\bar{s}') - V^{*,t-1}(s') \right| \\
&\leq \varepsilon_R + \gamma \frac{\max|R|}{1-\gamma} \varepsilon_T |\bar{S}| + \gamma \left( \varepsilon_R \gamma^{t-1} + \left( \varepsilon_R + \varepsilon_T \gamma \frac{\max|R|}{1-\gamma} |\bar{S}| \right) \sum_{i=0}^{t-2} \gamma^i \right) \\
&= \varepsilon_R + \varepsilon_T \gamma \frac{\max|R|}{1-\gamma} |\bar{S}| + \varepsilon_R \gamma^t + \left( \varepsilon_R + \varepsilon_T \gamma \frac{\max|R|}{1-\gamma} |\bar{S}| \right) \sum_{i=1}^{t-1} \gamma^i \\
&= \varepsilon_R \gamma^t + \left( \varepsilon_R + \varepsilon_T \gamma \frac{\max|R|}{1-\gamma} |\bar{S}| \right) \sum_{i=0}^{t-1} \gamma^i .
\end{aligned}$$

Using the subadditivity of the maximum and replicating the same derivations, we prove the induction step for the finite horizon case.

By taking the limit for $t \to \infty$ and $t = h$ in (A.8) in the infinite and finite horizon respectively, we obtain the claim. □

*Proof. [Theorem 5].* By the triangular inequality,

$$V^*(s) - V^{\bar{\pi}^*}(s) \leq \left| V^*(s) - \bar{V}^*(\bar{s}) \right| + \left| \bar{V}^*(\bar{s}) - V^{\bar{\pi}^*}(s) \right|$$

Given that $\bar{V}^* = \bar{V}^{\bar{\pi}^*}$, we can apply Lemma 2 to the second term on the right-hand side. On the other hand, Lemma 3 can be applied to the first term to derive the claim. □

We conclude this section with a brief remark on the error terms $\varepsilon_R$ and $\varepsilon_T$, which quantify the extent to which co-aggregated states share similar rewards and transition probabilities to other abstract states. These parameters intrinsically depend on both the structure of the underlying MDP and the aggregation function. To obtain the tightest and most informative bound on value loss, these constants should be chosen as small as possible.

**Observation 2.** By the definitions of the approximation errors introduced in the assumptions (A.5) and (A.4) and using the triangular inequality, we obtain

$$|R(s_1,a) - R(s_2,a)| \leq |\bar{R}(\bar{s},a) - R(s_1,a)| + |\bar{R}(\bar{s},a) - R(s_2,a)| \leq 2\varepsilon_R$$

for any $\bar{s} \in \bar{S}$, for any $s_1, s_2 \in \bar{s}$ and for any action $a \in A$, regardless of the specification of $\bar{R}$. Taking the maximum over all such combinations yields the lower bound on $\varepsilon_R$ as

$$\varepsilon_R \geq \frac{1}{2} \max_{\bar{s}, s_1, s_2 \in \bar{s}, a} |R(s_1, a) - R(s_2, a)|$$

If the abstract reward $\bar{R}(\bar{s}, a)$ is chosen to lie within the range of the rewards of states in the abstract state $\bar{s}$, that is $\bar{R}(\bar{s}, a) \in [\min_{s \in \bar{s}} R(s, a), \max_{s \in \bar{s}} R(s, a)]$, then this minimum is attained, and the tightest possible bound is achieved by setting

$$\varepsilon_R = \frac{1}{2} \max_{\bar{s}, s_1, s_2 \in \bar{s}, a} |R(s_1, a) - R(s_2, a)|$$

A similar reasoning applies to the transition error $\varepsilon_T$. If the abstract transition probability $\bar{T}(\bar{s}' \mid \bar{s}, a)$ is defined to lie within the range of transition probabilities from states in $\bar{s}$ to $\bar{s}'$, i.e. $\bar{T}(\bar{s}' \mid \bar{s}, a) \in \left[\min_{s \in s, a} \sum_{s' \in \bar{s}} T(s' \mid s, a), \max_{s \in s, a} \sum_{s' \in \bar{s}} T(s' \mid s, a)\right]$, then the minimum error value is given by

$$\varepsilon_T = \frac{1}{2} \max_{\bar{s}', \bar{s}, s_1, s_2 \in \bar{s}, a} \left| \sum_{s' \in \bar{s}'} T(s' \mid s_1, a) - \sum_{s' \in \bar{s}'} T(s' \mid s_2, a) \right|$$

## A.4. Tightness of the bound: a constructive example

We conclude this work by demonstrating that the value loss bound from Theorem 5 is tight in the worst-case sense. To do so, we construct a simple MDP and an associated abstraction for which the value loss matches the miminum of the upper bound.

We consider an MDP with four states $S = \{s_1, s_2, s_1', s_2'\}$ and two actions $A = \{0, 1\}$. We consider an aggregation function mapping the ground states into an abstract state space consisting of two abstract states $\bar{S} = \{\bar{s}, \bar{s}'\}$. Specifically, $\bar{s}$ is the abstract state grouping $s_1$ and $s_2$, that is $s_1, s_2 \in \bar{s}$, while $s_1'$ and $s_2'$ correspond to the abstract state $\bar{s}'$, that is $s_1', s_2' \in \bar{s}'$. A visual representation of this abstraction is shown in Figure A.1. The $s_1'$ and $s_2'$ are absorbing states, meaning that once either state is reached, the process remains there indefinitely regardless of the action taken. The reward function assigns $R(s_1', a) = 0$ and $R(s_2', a) = 1$ for any action $a \in A$. The transitions from $s_1$ and $s_2$ are deterministic and defined as

$$T(s_1' \mid s_1, a = 0) = T(s_1' \mid s_2, a = 1) = 1$$
$$T(s_2' \mid s_1, a = 1) = T(s_2' \mid s_2, a = 0) = 1$$

Rewards are defined such that any transition leading to $s_2'$ yields a reward of 1, while transitions to $s_1'$ yield a reward of 0.

Since the aggregated states have identical transitions to other abstract states, while $s_1$ and $s_2$– as well as $s_1'$ and $s_2'$– differ in their rewards by 1 for every available action, Observation 2 implies that the error terms can be set to

$$\varepsilon_T = 0 \qquad \varepsilon_R = \frac{1}{2}$$

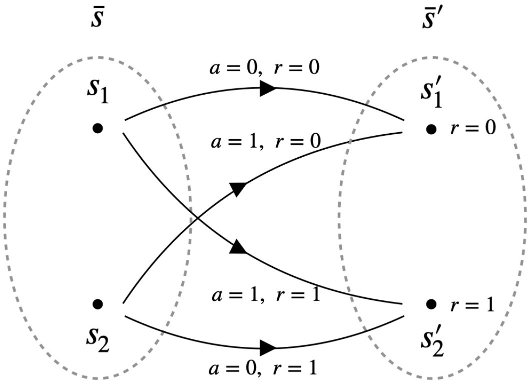


Figure A.1.: MDP example illustrating tightness of the value loss bound. Black dots represent ground states, arrows indicate deterministic transitions labeled with the corresponding actions and rewards, and dashed ovals denote abstract states.

Thus, the upper bound values are

$$\begin{aligned} \text{bound}_\infty &= \frac{1}{1-\gamma} && \text{infinite horizon} \\ \text{bound}_h &= h+1 && \text{finite horizon} \end{aligned}$$

The optimal policy $\pi^*(s_1) = 1$ and $\pi^*(s_2) = 0$ achieves for any state $s$ the value defined as

$$V^*(s) = \begin{cases} \sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma} & \text{infinite horizon} \\ \sum_{t=0}^{h} 1 = h+1 & \text{finite horizon} \end{cases}$$

However, an optimal abstract policy $\bar{\pi}^*$ must assign the same action to both $s_1$ and $s_2$ as they lie in the same abstract state $\bar{s}$. If it selects action 1, then $s_2$ transitions to $s_1'$ with reward 0; if it selects action 0, then $s_1$ transitions to $s_1'$ with reward 0. In either case, one of the two states will receive zero value, resulting in a value loss equal to the full value $V^*(s)$. Hence, for such state the bound is matched as

$$V^*(s) - V^{\bar{\pi}^*}(s) = V^*(s) = \begin{cases} \frac{1}{1-\gamma} & = \text{bound}_\infty \quad \text{infinite horizon} \\ h & = \text{bound}_h \quad \text{finite horizon} \end{cases}$$

This example proves that the bound is tight in the worst-case sense. However, it does not imply that the bound cannot be improved in general. Specifically, the constructed scenario corresponds to a specific case where the term of the upper bound related to the transition error vanishes, since $\varepsilon_T = 0$. As such, the example demonstrates that while the transition error term may be further tightened in general settings, the reward error term is, in that sense, unimprovable.

# REFERENCES

[1] R. Dearden and C. Boutilier. “Abstraction and approximate decision-theoretic planning”. In: *Artificial Intelligence* (1997), pp. 219–283.

[2] L. Li, T. J. Walsh, and M. L. Littman. “Towards a unified theory of state abstraction for MDPs”. In: *International Symposium on Artificial Intelligence and Mathematics (AI&Math)*. 2006.

[3] D. Abel, D. Hershkowitz, and M. Littman. “Near optimal behavior via approximate state abstraction”. In: *International Conference on Machine Learning (ICML)*. 2016.

[4] B. Van Roy. “Performance loss bounds for approximate value iteration with state aggregation”. In: *Mathematics of Operations Research* (2006), pp. 234–244.

[5] R. A. Howard. *Dynamic programming and Markov processes*. MIT Press, 1960.

# B

# Appendix of chapter 2

## B.1. Loss bounds proofs

In this section, we provide the formal proofs of the results presented in Section 3.5. We use the notation and terminology introduced in Section 3.3.

**Theorem 2.** Consider the two IALMs $\mathscr{M}$ and $\hat{\mathscr{M}}$, the following loss bound holds

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \leq 2h^2|R| \max_t \max_{\bar{s}^t, a^t} \|(T(\cdot \mid \bar{s}^t, a^t) - \hat{T}(\cdot \mid \bar{s}^t, a^t)\|_1$$

*Proof.* We apply the result for MDPs with uncertainty in transition probabilities presented in Corollary 2 of Mastin and Jaillet [1] to the exact and approximate IALM. Specifically, Corollary 2 states that given a finite-horizon MDP $M = (S, A, R, T, h)$ and an approximate MDP with uncertain transitions $\hat{M} = (S, A, R, \hat{T}, h)$, the performance loss when employing the optimal policy $\hat{\pi}^*$ for the approximate model $\hat{M}$ in the ground true model $M$ is bounded as

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \leq 2h^2|R| \max_{s,a} \|T(\cdot \mid s, a) - \hat{T}(\cdot \mid s, a)\|_1$$

We can apply this result by considering the IALM $\hat{\mathscr{M}}$ as an approximate version of the ground true IALM $\mathscr{M}$ with uncertain transitions $\hat{T}$. Then it suffices to consider the maximum over the time steps to obtain the claim. $\square$

**Lemma 1.** Consider the IALMs transitions $T$ and $\hat{T}$. For any $t$, augmented state $\bar{s}^t = (x^t, d^t) = ((x^t_{\text{int}}, x^t_{\text{dest}}), d^t)$ and action $a^t$,

$$\|(T(\cdot \mid \bar{s}^t, a^t) - \hat{T}(\cdot \mid \bar{s}^t, a^t)\|_1 \leq \|(I^t(\cdot \mid d^t) - \hat{I}^t(\cdot \mid d^t)\|_1 \qquad \text{(B.1)}$$

*Proof.* According to the expression for the IALM transitions (3.1), we have

$$
\begin{aligned}
|T(\bar{s}^{t+1}\mid \bar{s}^t, a^t) - \hat{T}(\bar{s}^{t+1}\mid \bar{s}^t, a^t)| &= T(x^{t+1}_{\text{int}}|x^t, a^t) \\
&\times \left|\sum_{y^t_{src}} T(x^{t+1}_{\text{dest}}\mid x^t, y^t_{src}, a^t)\left(I^t(y^t_{src}\mid d^t) - \hat{I}^t(y^t_{src}\mid d^t)\right)\right| \\
&\le T(x^{t+1}_{\text{int}}\mid x^t, a^t) \\
&\times \sum_{y^t_{src}} T(x^{t+1}_{\text{dest}}\mid x^t, y^t_{src}, a^t)\left|I^t(y^t_{src}|d^t) - \hat{I}^t(y^t_{src}|d^t)\right|
\end{aligned}
$$

Therefore considering the sum over $\bar{s}^{t+1}$, we obtain

$$
\begin{aligned}
&\|(T(\cdot\mid \bar{s}^t, a^t) - \hat{T}(\cdot\mid \bar{s}^t, a^t)\|_1 = \sum_{\bar{s}^{t+1}} |T(\bar{s}^{t+1}\mid \bar{s}^t, a^t) - \hat{T}(\bar{s}^{t+1}\mid \bar{s}^t, a^t)| \\
&\le \sum_{x^{t+1}_{\text{int}}, x^{t+1}_{\text{dest}}} T(x^{t+1}_{\text{int}}|x^t, a^t)\sum_{y^t_{src}} T(x^{t+1}_{\text{dest}}\mid x^t, y^t_{src}, a^t)|I^t(y^t_{src}\mid d^t) - \hat{I}^t(y^t_{src}\mid d^t)| \\
&= \sum_{y^t_{\text{src}}} \left|I^t(y^t_{\text{src}}\mid d^t) - \hat{I}^t(y^t_{\text{src}}\mid d^t)\right| \sum_{x^{t+1}_{\text{int}}} T(x^{t+1}_{\text{int}}\mid x^t, a^t)\sum_{x^{t+1}_{\text{dest}}} T(x^{t+1}_{\text{dest}}\mid x^t, y^t_{\text{src}}, a^t) \\
&= \sum_{y^t_{\text{src}}} \left|I^t(y^t_{\text{src}}\mid d^t) - \hat{I}^t(y^t_{\text{src}}|d^t)\right| \\
&= \left\|I^t(\cdot\mid d^t) - \hat{I}(\cdot\mid d^t)\right\|_1
\end{aligned}
$$

□

**Theorem 3.** Consider a IALM $\mathcal{M} = (S, A, T, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathcal{M}} = (S, A, \hat{T}, R, h, b^0)$. Then, a value loss bound in terms of the $L^1$-norm error is given by

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \le 2h^2|R|\max_{t,d^t}\left\|(I^t(\cdot\mid d^t) - \hat{I}(\cdot|d^t)\right\|_1 \tag{B.2}$$

*Proof.* It suffices to apply consecutively Theorem 2 and Lemma 1 to derive

$$
\begin{aligned}
\|V^* - V^{\hat{\pi}^*}\|_\infty &\le 2h^2|R|\max_t\max_{\bar{s}^t, a^t}\left\|T(\cdot\mid \bar{s}^t, a^t) - \hat{T}(\cdot\mid \bar{s}^t, a^t)\right\|_1 \\
&\le 2h^2|R|\max_t\max_{d^t}\left\|(I^t(\cdot\mid d^t) - \hat{I}^t(\cdot\mid d^t)\right\|_1
\end{aligned}
$$

□

**Corollary 1.** Consider a IALM $\mathcal{M} = (S, A, T, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathcal{M}} = (S, A, \hat{T}, R, h, b^0)$. Then, a value loss bound in terms of KL divergence error is given by

$$\|V^* - V^{\hat{\pi}^*}\|_\infty \le 2h^2|R|\max_{t,d^t}\sqrt{2D_{KL}(I^t(\cdot\mid d^t)||\hat{I}^t(\cdot\mid d^t))} \tag{B.3}$$

*Proof.* It suffices to apply Pinsker inequality [2] to equation (B.2) to obtain the claim

$$\begin{aligned}\|V^* - V^{\hat{\pi}^*}\|_\infty &\leq 2h^2|R|\max_{t,d^t}\left\|(I^t(\cdot\mid d^t) - \hat{I}(\cdot|d^t)\right\|_1\\ &\leq 2h^2|R|\max_{t,d^t}\sqrt{2D_{KL}(I^t(\cdot\mid d^t)||\hat{I}^t(\cdot\mid d^t))}.\end{aligned}$$

□

**Theorem 4.** Consider a IALM $\mathcal{M} = (S, A, T, R, h, b^0)$ and an AIP $\hat{I}$ inducing $\hat{\mathcal{M}} = (S, A, \hat{T}, R, h, b^0)$. Assume that for every time $t$ and d-set instantiation $d^t$, we have at least an influence sources sample of size $n$. Then for every $\varepsilon > 0$

$$\mathbb{P}\left(\|V^* - V^{\hat{\pi}^*}\|_\infty \leq 2h^2|R|\max_{t,d^t}\left\|I_n^t(\cdot\mid d^t) - \hat{I}^t(\cdot\mid d^t)\right\|_1 + \varepsilon\right) \geq 1 - h|D^h|(2^{|Y_{\text{src}}|} - 2)e^{-n\frac{\varepsilon^2}{2}} \tag{B.4}$$

where $|Y_{\text{src}}|$ is the cardinality of the influence sources space and $|D^h|$ the cardinality of the d-sets space at time $h$.

*Proof.* We use a result from Weissman *et al.* [3] that states that for a distribution $p$ over a space $S$ and an empirical distribution $p_n$ drawn from a sample of size $n$, for every $\varepsilon > 0$

$$\mathbb{P}(\|p - p_n\|_1 \geq \varepsilon) \leq (2^{|S|} - 2)e^{-n\frac{\varepsilon^2}{2}}$$

If we apply this result to the exact influence $I^t$ and the empirical influence $I_n^t$, we obtain

$$\mathbb{P}\left(\|I^t(\cdot\mid d^t) - I_n^t(\cdot\mid d^t)\|_1 \geq \varepsilon\right) \leq (2^{|Y_{\text{src}}|} - 2)e^{n\frac{\varepsilon^2}{2}},$$

Using the reverse triangle inequality

$$\left|\|I^t(\cdot\mid d^t) - \hat{I}^t(\cdot\mid d^t)\|_1 - \|\hat{I}^t(\cdot\mid d^t) - I_n^t(\cdot\mid d^t)\|_1\right| \leq \|I^t(\cdot\mid d^t) - I_n^t(\cdot\mid d^t)\|_1$$

in the previous inequality, we obtain

$$\begin{aligned}&\mathbb{P}\left(\|I^t(\cdot\mid d^t) - \hat{I}^t(\cdot\mid d^t)\|_1 \geq \|\hat{I}^t(\cdot\mid d^t) - I_n^t(\cdot\mid d^t)\|_1 + \varepsilon\right)\\ &\qquad\leq \mathbb{P}\left(\left|\|I^t(\cdot\mid d^t) - \hat{I}^t(\cdot\mid d^t)\|_1 - \|\hat{I}^t(\cdot\mid d^t) - I_n^t(\cdot\mid d^t)\|_1\right| \geq \varepsilon\right)\\ &\qquad\leq \mathbb{P}\left(\|I^t(\cdot\mid d^t) - I_n^t(\cdot\mid d^t)\|_1) \geq \varepsilon\right)\\ &\qquad\leq (2^{|Y_{\text{src}}|} - 2)e^{n\frac{\varepsilon^2}{2}}\end{aligned}$$

Thus by the union bound, we derive

$$
\begin{aligned}
\mathbb{P}\left(\max_{t,d^t}\|I^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1 \le \max_{t,d^t}\|I_n^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1+\varepsilon\right)&\\
\ge \mathbb{P}\left(\bigcap_{t,d^t}\{\|I^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1 \le \|I_n^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1+\varepsilon\}\right)&\\
=1-\mathbb{P}\left(\bigcup_{t,d^t}\{\|I^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1 \ge \|I_n^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1+\varepsilon\}\right)&\\
\ge 1-\sum_{t,d^t}\mathbb{P}\left(\|I^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1 \ge \|I_n^t(\cdot\mid d^t)-\hat{I}^t(\cdot\mid d^t)\|_1+\varepsilon\right)&\\
\ge 1-\sum_{t,d^t}(2^{|Y_{\text{src}}|}-2)e^{-n\frac{\varepsilon^2}{2}}&\\
\ge 1-h|D^h|(2^{|Y_{\text{src}}|}-2)e^{-n\frac{\varepsilon^2}{2}}&
\end{aligned}
$$

Now, we conclude the proof by combining the last inequality with the result from Theorem 3. □

## B.2. Experimental setup and additional results

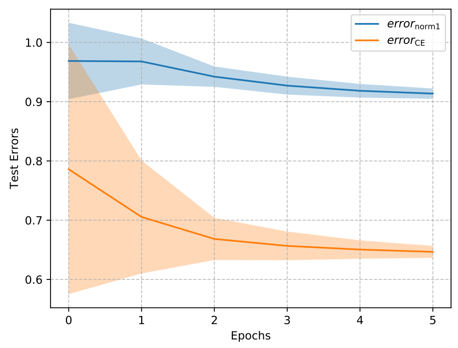
Here we describe the experimental setup for the domains we consider in Section 3.7 and collect some additional results for different scenario settings.

### B.2.1. Planetary exploratory

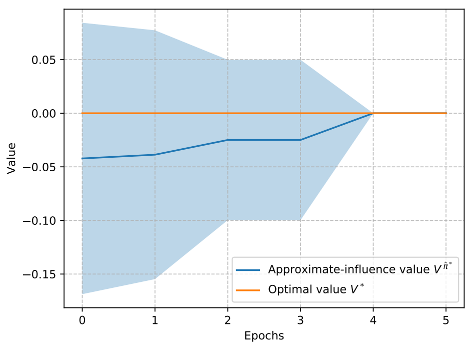
The setting is a version of the planetary exploration domain introduced by Witwicki and Durfee [4]. A planetary rover has to explore an area of an unknown planet. At the same time, a second agent, a satellite, may help the rover by providing a plan for the route. This increases the probability of a successful move by the rover. If the rover fails to move forward, it receives a penalty of −0.5, and reaching the target position results in a reward of +10. The satellite's actions are represented by a binary variable $a_{\text{sat}} \in \{0,1\}$ which represents if the satellite intends to provide a plan. The availability or non-availability of a routing plan is represented by a binary factor $pl \in \{\text{plan}, \neg\text{plan}\}$. The action $a_{\text{sat}}$ of the satellite depends on its own charge status, a resource which it has to manage as well. For the main results presented in the paper, the satellite employs a stochastic policy that corresponds to providing a plan whenever the level of its battery is higher than a given threshold. In this domain, we set the horizon to 6 and we will keep the initial state distribution fixed. We do not observe the charge, but we know that the history of the variable $pl$ constitutes the d-set for the influence sources $a_{\text{sat}}$. So we predict the influence $I^t(a^t_{\text{sat}} \mid (pl^0,\cdots,pl^t))$ at time $t$ based on the history of $pl$ up to time $t$.

We train a LSTM neural network model [5] over 15 training epochs with 10000 samples drawn from exploratory random policy of the rover. At each time step, the learning model receives as input the full history of plans $(pl^0,\cdots,pl^h)$ and predicts an approximate influence point $\{\hat{I}^t(\cdot\mid pl^0,\cdots,pl^t)\}_{t=0:h}$.

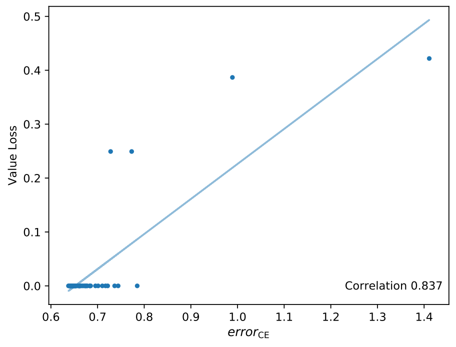
Figure B.1 and Figure B.2 show the value, test errors, and correlations for the rover's best response against two different policies of the satellite. Figure B.1 represents the scenation in which the satellite randomly decides whether to provide or not a plan. In Figure B.2 the satellite decides deterministically to provide a plan whenever the battery level is lower than a given threshold.



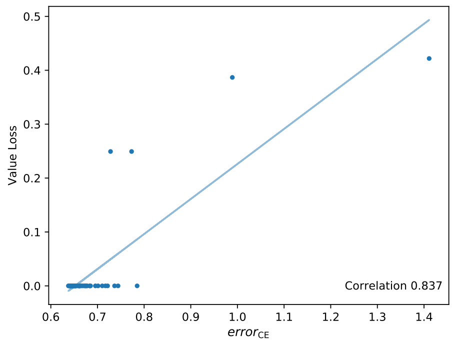
(a) Test errors for increasing training epochs.



(b) Value achieved by $\hat{\pi}^*$ compared to the optimal value for increasing training epochs.



(c) Correlation between value loss and $error_{\text{CE}}$.

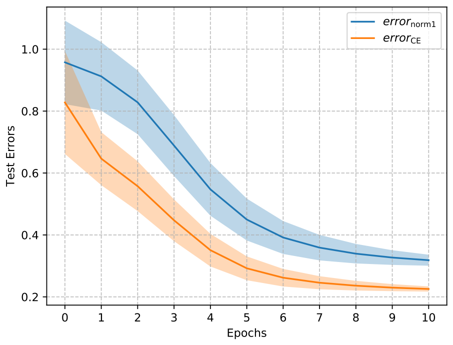


(d) Correlation between value loss and $error_{\text{norm1}}$.

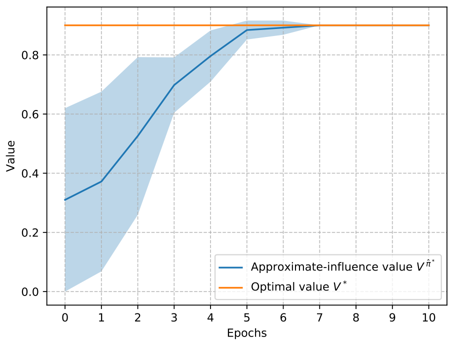
Figure B.1.: Random policy of the satellite.

### B.2.2. Traffic network

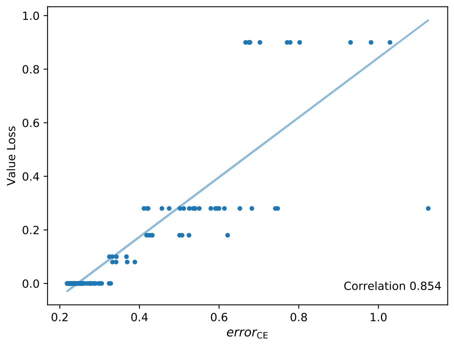
We simulate a busy traffic network with four lanes and four traffic light agents as depicted in Figure 3.2. At each time step, the agents observe the state of a $3 \times 3$ grid around their intersection. We consider the local agent to be the traffic light located at position $(1,1)$. We employ a fixed policy for the other three agents. Specifically, agents $(1,4)$ and $(4,1)$ prioritize the horizontal lane whenever there are vehicles waiting.



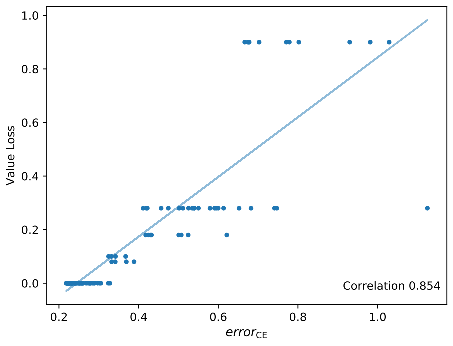
(a) Test errors for increasing training epochs.



(b) Value achieved by $\hat{\pi}^*$ compared to the optimal value for increasing training epochs.



(c) Correlation between value loss and $error_{\mathrm{CE}}$.



(d) Correlation between value loss and $error_{\mathrm{norm1}}$.

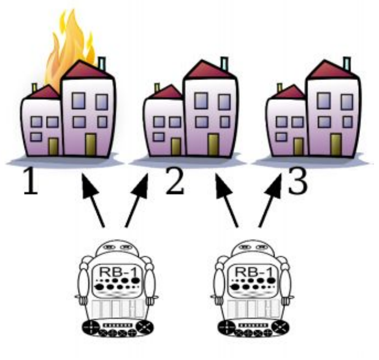
Figure B.2.: Deterministic policy of the satellite.

For agent (4,4), we use a random policy. The local space of the agent consists of observations from each side of the intersection, $x = (south, east, north, west)$, for $south, east, north, west \in \{0,1\}$, with 1 indicating the presence of a vehicle and 0 none. The local agent can decide to prioritize either the horizontal or vertical lane, $a_{loc} \in \{hor, vert\}$. It then receives a local reward corresponding to the number of cars waiting at the intersection, $R(x) = -east - north$. Vehicles departing from the west and south lanes have some probability of re-entering the network from a parallel lane, thereby affecting the decisions of other agents. In this domain, we set the horizon to 4 and keep the initial state distribution fixed. The only information the local agent needs to retrieve from the rest of the system, is whether a car will enter into the local model from the incoming lanes. Specifically, $X_{\mathrm{dest}} = (east, north)$. In order to predict the these variables, the agent uses the history of outgoing cars from $south$ and $west$, that is $D^t = ((south^0, east^0), \ldots, (south^t, east^t))$.

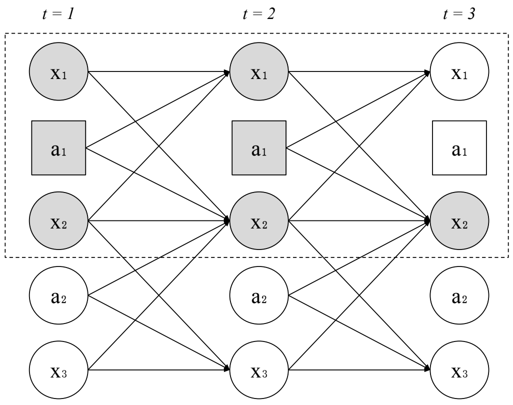
We train a LSTM model for 16 epochs with 10000 samples drawn from an exploratory random policy of the local agent. At each time step, the model sees the history of the $(south, east)$ and predicts the distribution of $(east^t, north^t)$.

### B.2.3. FIRE FIGHTERS PROBLEM

In the fire fighters problem introduced by Oliehoek [6], we model a team of two agents that have to extinguish the fire in a row of three houses. At each time step, an agent can choose to fight the fire at one of its neighboring houses as depicted in Figure B.3(a). The actions of the two agents are denoted by $a_1, a_2 \in \{0,1\}$, indicating whether the fire fighter chooses the left or right house, respectively. The state space



(a) An illustration of the fire fighters problem with two agents and three houses.



(b) A dynamic Bayesian network of the fire fighters domain. The dotted square refers to the local model of agent 1. The gray circles constitute the d-set for the influence sources $a_2, x_3$ at time $t = 3$.

Figure B.3.: Fire Fighters.

consists of three binary factors $x_1, x_2, x_3 \in \{0,1\}$, representing the state of the three houses, where 1 indicates a burning house. If a house is not burning and no fire fighter is present, the fire can spread from a neighboring house with probability 0.9. A single agent present at a house extinguishes the fire with probability 1 if none of the neighboring houses are burning, and with probability 0.6 otherwise. If two agents fight the fire at the same house, they extinguish it with probability 1. We consider a local model for agent 1, which can observe only its neighboring houses $x_1, x_2$ and receives a reward corresponding to the number of local houses that are burning, i.e., $R(s') = -x'_1 - x'_2$. Figure B.3(b) shows an illustration of the dynamic Bayesian network for this domain. We set the horizon to 4 and keep the initial state distribution fixed. The local agent does not observe the state of house $x_3$, therefore it cannot predict the action of the non-local agent $a_2$. However, the history of factors

$x_1, x_2$, action $a_1$ constitute the d-set for the influence sources $a_2, x_3$. So the agent predicts the influence $I^t(a_2^t, x_3^t \mid (x_1^1, a_1^1, x_2^1, \ldots, x_1^{t-1}, a_1^{t-1}, x_2^{t-1}, x_2^t))$ at time $t$.

We train a LSTM model up to 20 epochs with 100000 samples drawn from an exploratory random policy for the local agent 1. At each time step, the model takes as input the history of the houses $x_1, x_2$ and the local actions $a_1$ and predicts an approximate influence point $\{\hat{I}^t(\cdot \mid (x_1^1, a_1^1, x_2^1, \ldots, x_1^{t-1}, a_1^{t-1}, x_2^{t-1}, x_2^t))\}_{t=0:h}$.

# REFERENCES

[1] A. Mastin and P. Jaillet. “Loss bounds for uncertain transition probabilities in Markov decision processes”. In: *IEEE Conference on Decision and Control*. IEEE. 2012.

[2] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[3] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger. “Inequalities for the L1 deviation of the empirical distribution”. In: *Hewlett-Packard Labs, Tech. Rep* (2003).

[4] S. Witwicki and E. Durfee. “Influence-based policy abstraction for weakly-coupled Dec-POMDPs”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. 2010.

[5] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* (1997), pp. 1735–1780.

[6] F. Oliehoek. *Value-based planning for teams of agents in stochastic partially observable environments*. Amsterdam University Press, 2010.

# C
# Appendix of chapter 3

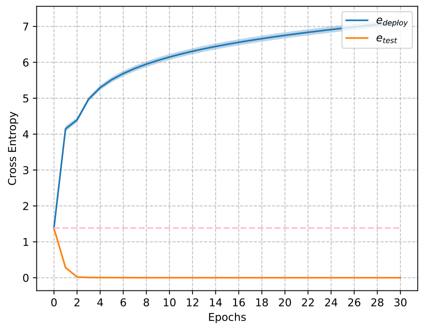
## C.1. Additional results

Figure C.1 and Figure C.2 show the test and deployment errors of the LSTM model for the influence over different training horizons in the grab a chair scenario with eleven agents (GC11) and traffic grid (TG) domain, respectively.

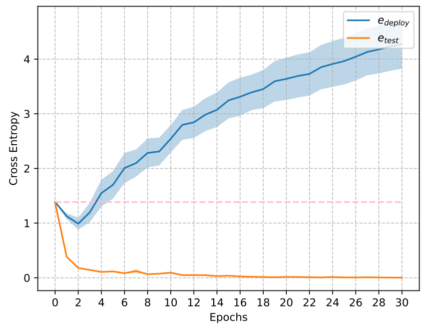
## C.2. Experimental domains

Here we describe in detail the experimental domains used for the empirical results, which are introduced in Section 4.3.3.

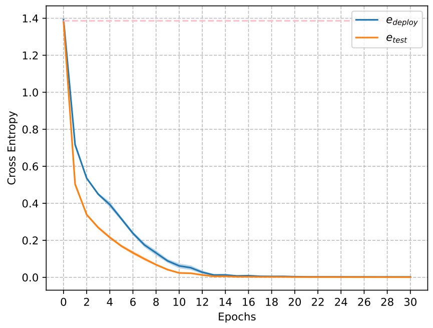
**Microgrid (MG).** This case study is inspired by previous works [1–3]. We model realistic interactions in an energy district (a microgrid) as a multi-agent problem. A hundred autonomous agents, referred to as *prosumer units*, manage the energy flexibility of the microgrid. Each agent decides whether to discharge stored energy or trade based on local demand and supply with the goal of minimizing the costs while ensuring energy balance and independence of the microgrid from the external power grid. Solar and wind energy sources are utilized. The problem focuses on a single local unit within a lattice network. The local state is given by, $s_{\text{local}} = (P_{RES}, P_d, SOC)$, where $P_{RES}$ is the renewable power production, $P_d$ is the stochastic power demand which is assumed to be normally distributed according to Hong and Fan [4], and the state of charge $SOC$ of the battery. The $P_{RES}$ includes solar and wind energy. The solar energy is modeled using hourly solar radiation data in `https://openweathermap.org/api/solar-radiation` and the photovoltaic power generation model introduced by Skoplaki and Palyvos [5]. The wind power generation is calculated by transforming the kinetic energy of the wind speed modeled via the Markov chain model described by Shamshad *et al.* [6], and assuming linear relationships with the power produced [7]. The energy produced by an agent may be used to meet the demand $P_d$, or stored in a battery to increase the $SOC$. At any time step $\Delta_t$ (one hour), the agents may decide to discharge the stored energy to meet the demand $P_d$, or trade energy with neighboring units. When buy and sell orders match, the power from the batteries is exchanged at a small cost/revenue for the buying and selling agents, respectively. After the local trading is cleared, to satisfy the power balance of the



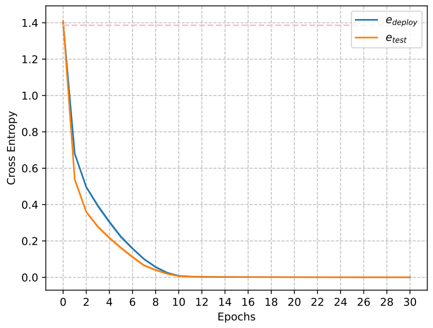
(a) $h_{\text{train}} = 2$



(b) $h_{\text{train}} = 4$



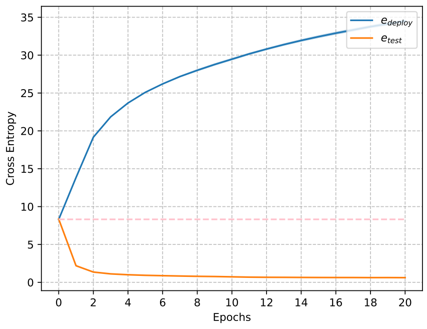
(c) $h_{\text{train}} = 6$



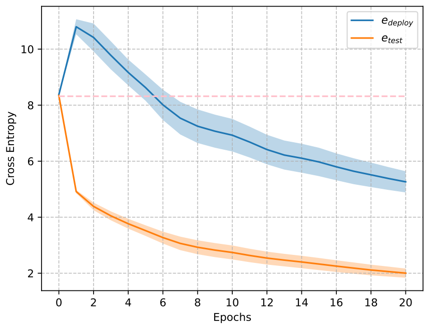
(d) $h_{\text{train}} = 8$

Figure C.1.: Deployment and test errors for LSTM in GC11 scenario. The dashed line represent the baseline accuracy of the random classifier.

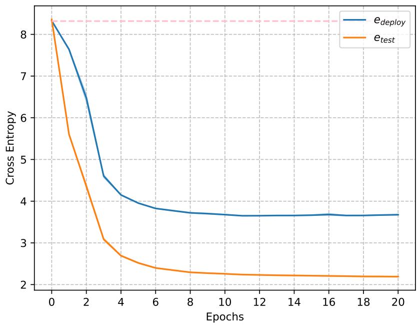
single units, every agent is forced to buy residual power from an external power distribution grid. A cost $C_{\text{ext}}$ is assigned to the energy not supplied ENS $= (P_d - P_{\text{deployed}}) \cdot \Delta_t$, where $P_{\text{deployed}}$ corresponds to the sum of the renewable power deployed and the power discharged from the battery. The cost $C_{\text{ext}}$ for buying from the external grid is much higher than the fixed operational costs of internal trade $C_{\text{int}}$. A schematic representation of a MG unit is depicted in Figure C.3(a). The individual reward for each agent is modeled as the sum of the cost/income for the internal trade (if any) and the negative costs of buying the energy not supplied from the external grid (if any) $r = \pm C_{\text{int}} \mathbb{1}_{\text{trade}} - C_{\text{ext}}\text{ENS}$. Thus, the team of agents shares the common objective to manage local resources to minimize the electricity costs constrained to satisfying the energy balance, generation limits and storage capacity. We take the perspective of a single unit in a lattice network highlighted in red in Figure C.3(b). The initial distribution of the battery is uniformly sampled at random and we consider $h = 40$ hours as the horizon of the problem. We assume that all the other agents act by storing or trying to buy power when the storage is scarce and discharging or trying to sell when power is abundant. Note that although the problem



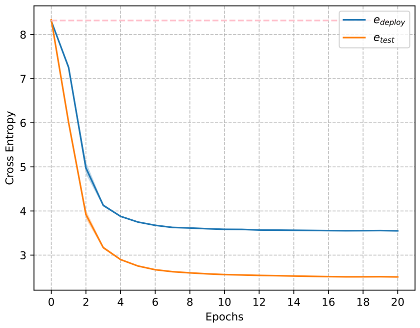
(a) $h_{\text{train}} = 10$



(b) $h_{\text{train}} = 20$



(c) $h_{\text{train}} = 30$

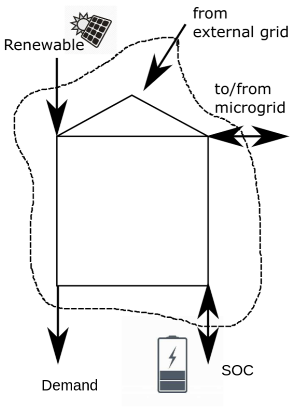


(d) $h_{\text{train}} = 40$

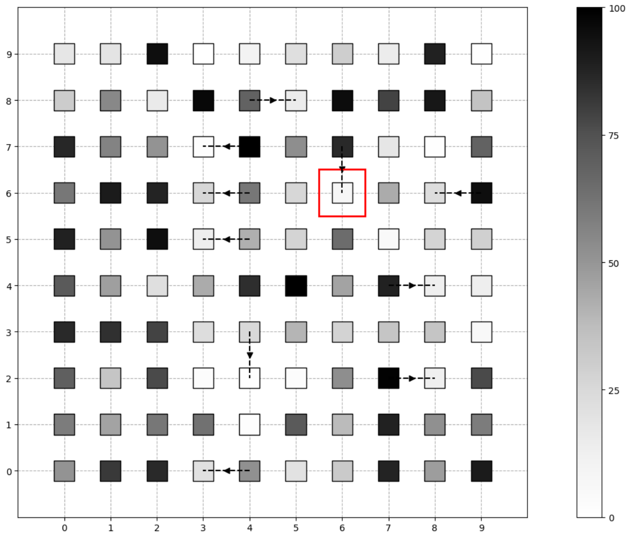
Figure C.2.: Deployment and test errors for LSTM in traffic grid domain. The dashed line represent the baseline accuracy of the random classifier.

size can be arbitrarily large, the influence experienced by the local agent only directly depends on the neighboring nodes in the network. Precisely, the only relevant information on the external portion of the system that an agent needs is whether the neighboring north, west, south and east agents will decide to sell or buy power. Although the distributions of influence sources $s_{src} = (a_{\text{N}}, a_{\text{W}}, a_{\text{S}}, a_{\text{E}})$ are affected (indirectly) by all the agents in the microgrid, the history of local actions $a$ provides sufficient statistics to predict the influence sources. The resulting problem consists of finding a function approximator for $I(a_{\text{N}}^t, a_{\text{W}}^t, a_{\text{S}}^t, a_{\text{E}}^t \mid a^0, \ldots, a^{t-1})$.

**Traffic grid (TG).** In this implementation of a traffic network, as described in Section 4.1 and 4.2, we simulate the vehicle traffic in a 9-intersections grid, schematically represented in Figure C.4. The sensors of each traffic light capture the vehicles in the $5 \times 5$ local grid at each intersection. The local model is represented as a red square for the selected protagonist agent. The other traffic lights employ hand-coded policies that prioritize lanes with higher car volumes. At time $t = 0$ the grid is initially empty, i.e., the initial



(a) Unit controlled by a single agent in the micro-grid. The local variables observed by the agent include the state of the charge, the power produced by renewable sources and the power demand.



(b) Multi-agent low-voltage grid. For each unit in the lattice, the state of charge (in percentage) is represented by the gray scale. Directed edges represent a power exchanged between prosumers (agents).

Figure C.3.: Microgrid.

state encodes no cars in the network. At each time step, a vehicle enters the network with a certain probability. The horizon is set to $h = 100$. The state of the environment is represented by binary state variables that indicate the presence or absence of a car at a given location in the grid. The goal of the agent is to minimize the total number of vehicles waiting at the local intersection. That is, the reward corresponds to the negative number of cars in the local model. To act optimally, the local agent needs to predict wether there will be incoming cars from the north end $s_{\mathrm{n}\downarrow}$ and the east end $s_{\mathrm{e}\leftarrow}$. Moreover, the local dynamics is affected by traffic congestion at intersection 2 and 4, as traffic jams can prevent vehicles from exiting the local model from the west and south ends. Therefore, the state variables for the outgoing ends and the actions of agents 2 and 4 are included in the set of influence sources. Thus, in addition to the factors encoding car inflows, the influence sources encompass four variables for the west outflow $s_{\mathrm{w}\leftarrow}$, four variables for the south outflow $s_{\mathrm{s}\downarrow}$, and the action $a_2$ and $a_4$, resulting in $s_{\mathrm{src}} = (s_{\mathrm{n}\downarrow}, s_{\mathrm{e}\leftarrow}, s_{\mathrm{w}\leftarrow}, s_{\mathrm{s}\downarrow}, a_2, a_4)$. The local information necessary to predict the influence sources includes the entire collection of local variables and actions. The influence that the agent predicts is expressed

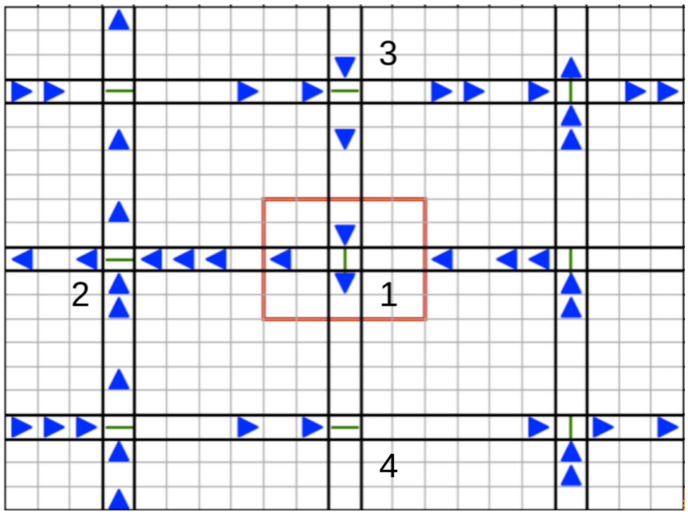


Figure C.4.: Traffic grid. The local model is delimited by the red square. The blue triangles represent the vehicles and the green bars the traffic lights.

by $I(s^t_{\mathrm{n}\downarrow}, s^t_{\mathrm{e}\leftarrow}, s^t_{\mathrm{w}\leftarrow}, s^t_{\mathrm{s}\downarrow}, a^t_2, a^t_4 \mid s^0_{\mathrm{local}}, a^0_1, \ldots, a^{t-1}_1, s^t_{\mathrm{local}})$.

**System admin (SA).** We use a multiagent version of the system administrator domain from Poupart and Boutilier [8]. A team of system administrators is responsible for the upkeep of a network of machines. Each node has a probability of failing at any time step, which increases when a neighboring machine in the network is down. Each agent only

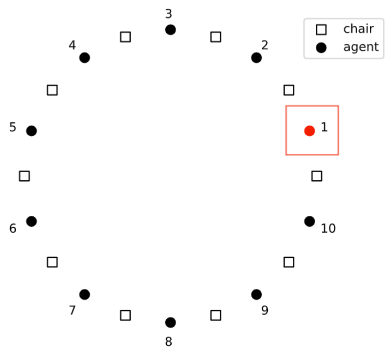


Figure C.5.: Grab a chair.

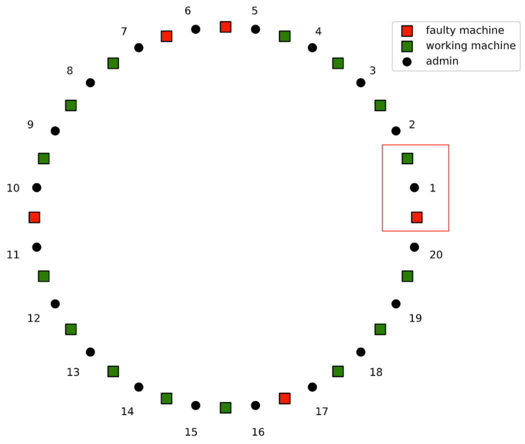


Figure C.6.: System admin.

observes the status of the machines in its proximity. Consequently, it may decide to intervene by trying to reboot the system of one of these nodes. With a certain probability, the process will succeed, resulting in a working node at the next time step. When more than one agent decides to reboot one machine, the process succeeds with probability 1. The goal of the team is to secure the highest number of working machines. Specifically, each agent receives a penalty for each faulty machine under its control. We consider a network of $N = 20$ machines organized in a ring configuration as depicted in Figure C.6. Each admin agent $i$ is responsible for the maintenance of two neighboring nodes whose states, denoted by $x_i$, $x_{i+1}$, are fully observable. We take the perspective of a single administrator, for instance agent 1 in Figure C.6, whose local model includes only the states (faulty or working) of the two neighboring machines $s_{\text{local}} = (x_1, x_2)$ and its action $a_1$. The problem horizon is set to $h = 500$ time steps, and initially a random state is sampled for each machine. To act optimally, agent 1 needs to determine whether agents 2 and 20 will decide to reboot one of the two machines they share control over. Also, it must consider the neighboring machine states $x_3$, $x_{20}$ as they may increase the probability of failures of local machines. According to the influence formalism introduced in Section 4.2, the sources of influence correspond to $s_{\text{src}} = (x_3, x_{20}, a_2, a_{20})$. The local information available to the agent 1 to predict the sources of influence consists of the entire collection of local variables, that is, $(x_1, x_2, a_1)$. Thus, the influence problem consists of finding an approximation for the distribution $I(x_3^t, x_{20}^t, a_2^t, a_{20}^t \mid x_1^0, x_2^0, a_1^0, \ldots, a_1^{t-1}, x_1^t, x_2^t)$.

**Grab a chair (GC).** In this simplified version of the SA problem, introduced by He, Suau de Castro, and Oliehoek [9], $N$ agents arranged in a ring decide at each time step whether to grab the chair on their left or right side, as shown in Figure C.5. An agent successfully obtains the chai–and thus receives a rewar–only if the neighboring agent has not targeted the same chair. After taking an action, each agent only observes whether it managed to grab the chair, ignoring the action of the neighboring agents. The local agent, numbered by 1 and depicted in red in Figure C.5, has access only to its own actions and rewards, which form the local model. The horizon is set to $h = 200$, and initially, each agent deterministically chooses the chair on its right side. After that, all non-local agents act by copying the previous action of the following agent in counterclockwise order. For the decision-making problem of the local agent, the only information required to act optimally consists of the decisions of its neighboring agent 2 and $N$ as they directly influence the possibility to secure a chair. Contrarily, the other agents affect the local model only indirectly. Consequently, the local agent needs to predict the influence sources corresponding to the actions $s_{\text{src}}^t = (a_2^t, a_N^t)$ given the local information of the d-set $d_{\text{set}}^t = (a_1^0, \ldots, a_1^{t-1})$. Thus, the influence to predict corresponds to $I(a_2^t, a_N^t \mid a_1^0, \ldots, a_1^{t-1})$.

## C.3. VISUALIZATION OF THE PREDICTIONS

Short videos showcasing the influence predictions for the traffic grid and grab a chair are available here. In the traffic grid video, the left side displays the global model simulator, while the right side shows the local model alongside the influence source predictions of an LSTM trained over 30 time steps, achieving a cross entropy test error of approximately 3.7 (corresponding to TG in Table 4.2). The predicted probability distribution over the binary variables, which represent the presence of a car, is visualized using the color intensity: darker colors indicate predictions closer to 1. In this scenario most influence sources–specifically, those corresponding to the traffic outflows from the local model–exhibit predominantly deterministic behavior, meaning their values are almost entirely determined by the local variables. This provides a visual intuition of the quality of the predictions, which appear to be quite accurate. In the grab a chair video, we visualize the predictions of an LSTM trained over 6 steps by the blue markers, together with the global simulation. Aligned with the cross entropy test error being close to zero (see GC4 in Table 4.2), the predictions appear to match perfectly the deterministic values of the influence sources.

## C.4. IMPLEMENTATION DETAILS

In the following tables, we collect the implementation details.

| **Domain** | $h$ | **#Agents** | **Policies** | $b_0$ | **#Inf Sources** | **D-set dim** |
|---|---|---|---|---|---|---|
| Microgrid | 40 | 100 | ranges | uniform | 4 | 1 |
| Traffic grid | 100 | 9 | priority | zeros | 12 | 9 |
| System admin | 500 | 20 | mixed | uniform | 4 | 3 |

Table C.1.: Setting of the scenarios and features of the influence learning tasks.

| | **Optimization** | | | | |
|---|---|---|---|---|---|
| **Domain** | **Sample size** | **Batch size** | **#Epochs** | **Alg** | **LR Decay** |
| Microgrid | 500 | 100 | 15 | Adam | Linear |
| Traffic grid | 500 | 100 | 20 | Adam | Linear |
| System admin | 500 | 100 | 20 | Adam | Linear |

Table C.2.: Optimization hyperparameters for the learning models.

| **Domain** | $H$ | $h_{\text{deploy}}$ | **#Agents** | **Policies** | $b_0$ |
|---|---|---|---|---|---|
| GC4 | $\{2,\ldots,14\}$ | 200 | 4 | copy | right chair |
| GC11 | $\{5,\ldots,30\}$ | 200 | 11 | copy | right chair |
| TG | $\{10,\ldots,100\}$ | 500 | 9 | priority | zero vehicles |

Table C.3.: Setting of the scenarios for long horizon tasks.

| **Domain** | **Batch size** | **#Epochs** | **LR Init** | **LR Final** | **LR Decay** | ***n*** | ***m*** |
|---|---|---|---|---|---|---|---|
| GC4 | 10 | 25 | $10^{-2}$ | $10^{-5}$ | linear | 500 | 100 |
| GC11 | 10 | 30 | $10^{-2}$ | $10^{-5}$ | linear | 500 | 100 |
| TG | 10 | 20 | $10^{-2}$ | $10^{-5}$ | linear | 500 | 100 |

Table C.4.: Optimization choices for long horizon tasks.

| | | **Architecture** | | | | | |
|---|---|---|---|---|---|---|---|
| **Domain** | **Model** | **#Layers** | **#Units** | **Kernel** | **#Param** | **Activ** | **Reg** |
| GC4 | LSTM | 1 | 10 | - | 564 | Tanh | None |
| | FullyConv | 4 | [6,6] | 4 | 544 | ReLU | Drop |
| GC11 | LSTM | 1 | 32 | - | 4612 | Tanh | None |
| | FullyConv | 8 | [10,10,10,10] | 6 | 4484 | ReLU | Drop |
| TG | LSTM | 1 | 32 | - | 2136 | Tanh | None |
| | FullyConv | 4 | [8,8] | 6 | 1944 | ReLU | Drop |

Table C.5.: Architectures of the learning models for long horizon tasks.

| | **Architecture** | | | | | |
|---|---|---|---|---|---|---|
| **Models** | **#Layers** | **#Units** | **Kernel** | **#Param** | **Activ** | **Reg** |
| (*size* ≤ *100*) | | | | | | |
| LSTM | 1 | 2 | - | 88 | Tanh | None |
| GRU | 1 | 2 | - | 78 | Tanh | None |
| TCN | 2 | 2,2 | 8 | 100 | ReLU | None |
| FullyConv | 2 | 1 | 8 | 52 | ReLU | Drop |
| (*size 1000*) | | | | | | |
| LSTM | 1 | 13 | - | 1056 | Tanh | None |
| GRU | 1 | 14 | - | 954 | Tanh | None |
| TCN | 4 | 6,6,6,6 | 8 | 1048 | ReLU | None |
| FullyConv | 4 | 6,6 | 8 | 1084 | ReLU | Drop |
| (*size 15000*) | | | | | | |
| LSTM | 1 | 56 | - | 14128 | Tanh | None |
| GRU | 1 | 64 | - | 13904 | Tanh | None |
| TCN | 5 | 20,20,20,20,20 | 8 | 13396 | ReLU | None |
| FullyConv | 8 | 15,15,15,15 | 8 | 13246 | ReLU | Drop |
| LogReg | - | - | - | 13104 | None | None |

Table C.6.: Microgrid. Architectures of the learning models.

| | Architecture | | | | | |
|---|---|---|---|---|---|---|
| **Models** | **#Layers** | **#Units** | **Kernel** | **#Param** | **Activ** | **Reg** |
| (*size* ≤*200*) | | | | | | |
| LSTM | 1 | 2 | - | 176 | Tanh | None |
| GRU | 1 | 2 | - | 150 | Tanh | None |
| TCN | 2 | 2,2 | 4 | 164 | ReLU | None |
| FullyConv | 2 | 2 | 4 | 188 | ReLU | Drop |
| (*size 1000*) | | | | | | |
| LSTM | 1 | 9 | - | 960 | Tanh | None |
| GRU | 1 | 11 | - | 1014 | Tanh | None |
| TCN | 4 | 4,4,4,4 | 10 | 976 | ReLU | None |
| FullyConv | 4 | 4,4 | 10 | 1032 | ReLU | Drop |
| (*size 10000*) | | | | | | |
| LSTM | 1 | 42 | - | 9936 | Tanh | None |
| GRU | 1 | 49 | - | 10020 | Tanh | None - |
| TCN | 4 | 16,16,16,16 | 10 | 9592 | ReLU | None |
| FullyConv | 4 | 16,16 | 10 | 9816 | ReLU | Drop |
| (*size 50000*) | | | | | | |
| LSTM | 1 | 104 | - | 50360 | Tanh | None |
| GRU | 1 | 120 | - | 50064 | Tanh | None |
| TCN | 6 | [30]x6 | 10 | 48624 | ReLU | None |
| FullyConv | 6 | [30,30,30] | 10 | 49104 | ReLU | Drop |
| (*size* 1M) | | | | | | |
| LogReg | - | - | - | 1093200 | None | None |

Table C.7.: Traffic grid. Architectures of the learning models.

| | **Architecture** | | | | | |
|---|---|---|---|---|---|---|
| **Models** | **#Layers** | **#Units** | **Kernel** | **#Param** | **Activ** | **Reg** |
| (*size* ≤ *100*) | | | | | | |
| LSTM | 1 | 2 | - | 80 | Tanh | None |
| GRU | 1 | 2 | - | 66 | Tanh | None |
| TCN | 2 | 2,2 | 4 | 68 | ReLU | None |
| FullyConv | 2 | 2 | 4 | 80 | ReLU | Drop |
| (*size 1000*) | | | | | | |
| LSTM | 1 | 12 | - | 920 | Tanh | None |
| GRU | 1 | 14 | - | 918 | Tanh | None |
| TCN | 4 | 6,6,6,6 | 8 | 1088 | ReLU | None |
| FullyConv | 4 | 6,6 | 8 | 1136 | ReLU | Drop |
| (*size 10000*) | | | | | | |
| LSTM | 1 | 48 | - | 10568 | Tanh | None |
| GRU | 1 | 54 | - | 9998 | Tanh | None |
| TCN | 6 | [16] x6 | 8 | 10856 | ReLU | None |
| FullyConv | 6 | 16,16,16 | 8 | 11016 | ReLU | Drop |
| (*size* 3M) | | | | | | |
| LogReg | - | - | - | 2.9M | None | None |

Table C.8.: System admin. Architectures of the learning models.

# ACKNOWLEDGMENTS

First, I want to thank my promotor Frans for selecting me for this position and for giving me the opportunity to work at this PhD project for all these years. When I think back to the humble, trusting, and compliant student that I was at the time of my first interview, I can comfortably say that remarkable changes have occurred. I thank my promotor Catholijn for the useful feedback on my Propositions.

An honest thank you goes to my office mate Aleks, who witnessed this journey from the very beginning. Thanks for the many discussions about the future and for filling an empty office with laughter, thoughts, shared worries, songs, and jokes during pandemics.

Tom, I am not sure I would have made it to the end of this process without your constant support and advice, but also without all the countless fun experiences we have had along the way. Whether we are sharing bike rides, worrying about Capstone projects, co-working, drinking beer (prosecco for me) or traveling, time always flies when you are around!

To my colleagues and friends, Zuza and Davide. I cannot imagine how the office days would have looked without you two. Zuza, thank you for being one of those who always asks personal questions. I feel really lucky to have had such a generous and caring person on my side during these years. Davide, thank you for being brave enough to aim for better for yourself. Whether a theoretical, coding, or education-related problem arises, you have always enthusiastically engaged in discussions and brought good ideas to the table. I also want to thank you both for joining in many fun plans. I wish we could be bartenders, movers, flatmates, hikers, party and travel buddies many more times together.

I want to thank my first office mates, Jinke and Rolf, for teaching me how to play ping pong through an endless series of losses, with no wins recorded (yet). Oussama, I truly appreciate your kind and reflexive attitude.

I would like to thank my colleagues from the teaching team for all the help and patience with me. I am really glad to work among colleagues who are so passionate and dedicated to our work.

I want to thank all my students, including those who tested my perseverance, for the opportunity to contribute to their education and for giving meaning to my daily efforts to grow as a teacher. A special thanks goes to my very first student, Victoria, the only one so far I have managed to convince to publish a paper out of her remarkable efforts.

I am so grateful for the many places in the world that welcomed me and offered me interesting office spaces, where each of my papers came to life. Thanks to all the beautiful people I have met along my adventurous paths and in the dance communities I have been part of. For enriching my life and reminding me how to stay human.

Here, I would like to thank all those who have been part of my large elective family.

Angelo, riesci sempre a strapparmi un sorriso dal cuore; grazie per aver voglia di ballare, correre e mangiare al cinese pezzotto con me. Grazie, Federica, per aver tenuto la tua porta di casa di Leiden aperta per me. Franco, per l'entusiasmo che hai messo in tutto quello che abbiamo fatto insieme, dagli ostelli alle barche, dalle cantate in strada alle lunghe vasche di quartiere, per tutti gli splendidi ricordi che ho della pandemia, grazie. Gaia, ti voglio ringrare per tutta la tua disponibilità ad aiutarmi e ad ascoltarmi; ma anche per le risate a crepapelle, per le splendide scemenze in cui ci seguiamo e per la tua trascinante voglia di fare. Maria, gracias por todas las buenas energías que has llevado en un momento tan gris. Por los viajes, los bailes, las risas y las lágrimas que compartimos y que aún vamos a compartir. Serena, la tua tenacia ci riporta periodicamente tutti intorno alla stessa tavola per sentirci di nuovo in famiglia; grazie per l'affetto incondizionato che mi hai regalato in tutti questi anni.

Vorrei ringraziare il mio amico più longevo, Andrea, per essersi divertito a realizzare la copertina di questa tesi con me e per ricordarmi da dove siamo venuti anni fa insieme. Claudia, grazie per avermi accompagnata per mano nei primi difficili anni di questo percorso. Giulia, senza quella telefonata, non sono certa che la mia vita avrebbe preso tutte queste bellissime e diverse direzioni. Alla mia dottoressa va un grande grazie per tutto l'affetto non dovuto che ha messo nella mia terapia.

Un grazie va a mia madre, Anna, per essere mia madre, che so non è affatto semplice, per essere aperta ad ascoltare, a cambiare prospettiva per raggiungere vette altissime. A mia sorella, Alberta, per essere stata una guida nelle decisioni critiche degli ultimi anni, per le tante volte in cui ha revisionato il mio lavoro, e per offrirmi, insieme a mio cognato Johnny, un posto dove avere voglia di tornare. A Lorenzo, mio nipote, per tutti gli interessanti spunti di riflessione che le nostre conversazioni offrono, per la tua curiosità, e la voglia di condividere con me giochi, gelati, indovinelli e il letto a castello.

The last acknowledgment, I owe it to myself. For all the times I managed to stand still and for all those times I fell. For the courage to assert myself and the courage to surrender. For having come to understand my worth.

# CURRICULUM VITÆ

## Elena CONGEDUTI

| | |
|---|---|
| 13-03-1989 | Born in Rome, Italy |

## EDUCATION

| | |
|---|---|
| 2009–2013 | Bachelor in Mathematics<br>La Sapienza, University of Rome |
| 2014 - 2016 | Mater in Mathematics<br>La Sapienza, University of Rome (2014 – 2015)<br>University of Leiden (2015–2016) |
| 2016–2017 | Master in Data Science<br>University of Bologna |
| 2018–2026 | Ph.D. in Computer Science<br>Delft University of Technology |

## WORK EXPERIENCE

| | |
|---|---|
| 2017–2018 | Data Scientist<br>Engineering Ingegneria Informatica |
| 2023–now | Lecturer in Computer Science<br>Delft University of Technology |

# List of Publications

Related to the Ph.D research

7. E. Congeduti, R. Rocchetta, and F. A. Oliehoek. “Influence Learning in Complex Systems”. In: *Transactions on Machine Learning Research (TMLR)* (2025)

6. R. Starre, M. Loog, E. Congeduti, and F. Oliehoek. “An analysis of model-based reinforcement learning from abstracted observations”. In: *Transactions on Machine Learning Research (TMLR)* (2023)

5. M. Suau, J. He, E. Congeduti, R. Starre, A. Czechowski, and F. Oliehoek. “Influence-aware memory architectures for deep reinforcement learning in POMDPs”. In: *Neural Computing and Applications* (2022), pp. 1–17

4. E. Congeduti and O. Frans. “A cross-field review of state abstraction for Markov decision processes”. In: *Benelux Conference on Artificial Intelligence (BNAIC) and Belgian Dutch Conference on Machine Learning (BeNeLearn)*. 2022

3. V. Catalán Pastor, E. Congeduti, and F. Oliehoek. “Overcoming traffic sensors malfunctions with deep learning”. In: *Benelux Conference on Artificial Intelligence (BNAIC) and Belgian Dutch Conference on Machine Learning (BeNeLearn)*. 2022

2. E. Congeduti, A. Mey, and F. Oliehoek. “Loss bounds for approximate influence-based abstraction”. In: *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2021

1. M. Suau, E. Congeduti, R. Starre, A. Czechowski, and F. Oliehoek. “Influence-based abstraction in deep reinforcement learning”. In: *AAMAS workshop on adaptive learning agents*. 2019

Not related to the Ph.D research

1. V. Bellelli, G. Siccardi, L. Conte, L. Celani, E. Congeduti, C. Borrazzo, L. Santinelli, G. P. Innocenti, C. Pinacchio, G. Ceccarelli, *et al.* “Preliminary attempt to predict risk of invasive pulmonary aspergillosis in patients with influenza: decision trees may help?” In: *Antibiotics* (2020), p. 644

# Learning Local Abstractions in Complex Sequential Decision-Making Systems

**Elena Congeduti**