Model-Based Reinforcement Learning with State Abstraction: A Survey

Rolf A. N. Starre[1], Marco Loog[2], and Frans A. Oliehoek[1]

[1] Delft University of Technology, Delft, The Netherlands
{r.a.n.starre, f.a.oliehoek}@tudelft.nl
[2] Radbout University, Nijmegen
marco.loog@ru.nl

Abstract. Model-based reinforcement learning methods are promising since they can increase sample efficiency while simultaneously improving generalizability. Learning can also be made more efficient through state abstraction, which delivers more compact models. Model-based reinforcement learning methods have been combined with learning abstract models to profit from both effects. We consider a wide range of state abstractions that have been covered in the literature, from straightforward state aggregation to deep learned representations, and sketch challenges that arise when combining model-based reinforcement learning with abstraction. We further show how various methods deal with these challenges and point to open questions and opportunities for further research.

Keywords: Model-based RL · State Abstraction · MDPs

1 Introduction

With roots in sequential analysis [77], Reinforcement Learning (RL) is a general framework for learning how to act near-optimally in sequential decision-making problems. A key challenge for RL is sample efficiency. Sample efficiency is important because, in many problems, it can be expensive, in time or monetary costs, to collect samples. The combination of Model-based Reinforcement Learning (MBRL) and abstraction is of interest for improving the sample efficiency of learning methods that aim to find solutions for sequential decision-making problems. We define MBRL as an RL method that explicitly learns a model of the environment. MBRL provides a way to find solutions to complex problems efficiently [71] and allows for transfer in shifting or related tasks [1, 53]. The state representation, the input to RL methods, plays an essential role in the learning process. A state representation will often contain irrelevant details, e.g., when the input is an image, a large amount can consist of a background that has no direct relevance to the task. Abstracting the state representation to remove irrelevant parts for optimal decision-making allows RL methods to learn much faster. Learning to decide which parts of the state representation are relevant is a key aspect of abstraction learning.

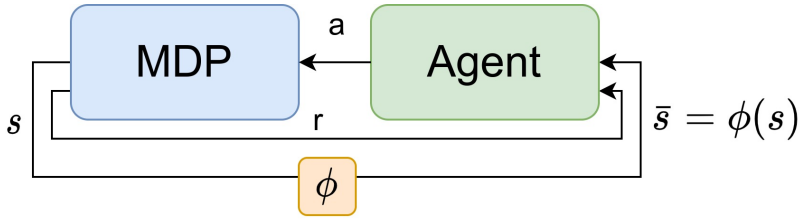

Fig. 1. RL with abstraction, the agent observes $\bar{s} = \phi(s)$ instead of s. Image from [70].

State abstraction can be carried out in various ways, ranging from state aggregation [2, 44] to deep learned representations [63, 66]. We provide a high-level view of the promising research in the field, covering a wide range of different types of state abstractions known from the literature.

Recently MBRL, abstraction learning, and related topics have received much attention. There are surveys of decision-making under uncertainty [36], MBRL in general [53], deep MBRL [62], and representation learning in both robotics [42] and MBRL [32]. Our work takes a broad view of abstraction and focuses on the additional challenges that arise when combining MBRL and abstraction [1, 57, 69, 70]. The contributions of this work are the following: We detail challenges that arise from the combination of MBRL with abstraction using the view of abstraction plus RL as a Partially Observable MDP (POMDP). W e show how different approaches for MBRL with state abstraction deal with these challenges, providing a unified view of a wide range of approaches in the process. We identify open questions and opportunities for further research.

2 An Overview of State Abstraction for RL

We consider RL in sequential decision-making problems, which can be defined as a Markov decision process (MDP) [64]: $\langle S, A, T, R, \gamma \rangle$, where S is a set of states $s \in S$, A a set of actions $a \in A$, T a transition function $T(s'|s, a) = \Pr(s'|s, a)$, R a reward function $R(s, a)$ which gives the reward received when the agent executes action a in state s, and γ the discount factor ($0 \leq \gamma < 1$). For realistic problems, the state space of the MDP representation is often too large to tackle directly. One way to reduce the size is to use compact representations such as state abstractions. Section 2.1 characterizes different state abstractions methods and briefly describes some of their properties. Section 2.2 describes how abstraction in an MDP can be viewed as a POMDP and the resulting challenge.

2.1 Characterization of Abstractions

State abstraction can be used to reduce the problem size by clustering states into abstract states. This clustering can be defined by using an abstraction function ϕ, which maps (or aggregates) ground states s to abstract states $\bar{s}$, where the

bar notation denotes objects in the abstract space. Here we consider a discrete state space and write this mapping as $\phi(s) = \bar{s}$, such that the abstract state space can be written as $\bar{S} = \{\phi(s) \mid s \in S\}$. The agent then uses the abstract states $\bar{s}$ and the rewards for learning transitions and rewards over the abstract state space. State abstraction can result in an abstract state space that is much smaller than the original state space, $|\bar{S}| \ll |S|$, which can make learning easier.

In the planning setting, where we have access to the model of a problem, many different abstraction functions have been considered [2, 44]. Abstractions group states based on specific criteria of the state or state-action pairs. An example is the (stochastic) bisimulation [21], also known as model-irrelevance abstraction [44]. In this abstraction, states are only grouped if their reward and transition functions in the abstract space are the same, i.e., $\phi(s_1) = \phi(s_2)$ iff

$$\forall_{a \in A}\ R(s_1, a) = R(s_2, a), \tag{1}$$

$$\text{and } \forall_{\bar{s}' \in \bar{S}}\ T(\bar{s}'|s_1, a) = T(\bar{s}'|s_2, a). \tag{2}$$

Here $T(\bar{s}'|s, a)$ is the transition to an abstract state $\bar{s}'$ which is defined as

$$T(\bar{s}'|s, a) \coloneqq \sum_{s' \in \bar{s}'} T(s'|s, a). \tag{3}$$

If we have access to the MDP, we can compute a more compact abstract MDP [13] and find a solution for this smaller problem. An important aspect of these abstractions is whether or not (near) optimal policies for the original policy can be obtained when the policy is learned from the abstract problem. Several results showing that this is possible have been obtained for multiple forms of abstraction [44, 2]. These results make abstractions interesting for RL as they show that it is possible to significantly reduce the problem size while still being able to obtain (near) optimal policies for the original problem.

To allow for further reduction in the problem size, approximate versions of abstractions, such as the ϵ-bisimulation, have been considered [2, 44]. In the approximate versions, the grouping criteria are relaxed. E.g., in the ϵ-bisimulation, the transition and reward functions for grouped states will be close but not necessarily the same, i.e., $\phi(s_1) = \phi(s_2)$ iff

$$\forall_{a \in A}\ |R(s_1, a) - R(s_2, a)| \leq \epsilon, \tag{4}$$

$$\text{and } \forall_{\bar{s}' \in \bar{S}}\ |T(\bar{s}'|s_1, a) - T(\bar{s}'|s_2, a)| \leq \epsilon, \tag{5}$$

where $T(\bar{s}'|s, a)$ is defined as in (3). Several other examples of exact and approximate state abstraction functions can be found in the literature [2, 44]. For a given MDP, it is possible to build an abstract MDP using ϵ-bisimulation criteria [14]. Recent work has introduced transitive state abstractions, which can be computed efficiently [1]. If we have a compact model, the goal is to find a good policy. A potential issue is that if a learned model only approximates the true model, minor errors can compound when planning for long horizons [73, 81]. Results for planning have shown that for particular approximate state representations, such as ϵ-bisimulation, the learned policy can still be approximately

optimal [2]. There is a similar result for using RL in an abstract MDP [72]. However, these results assume that we have access to the MDP or an abstract MDP, which requires the problem to be known, and this is typically not the case in RL.

2.2 Abstraction in an MDP as a POMDP

In the general case of MBRL in an unknown MDP with an abstraction ϕ, the situation will be as depicted in Figure 1. Without abstraction, the agent receives a state s as an observation. With abstraction, the agent instead observes an abstract state $\bar{s} = \phi(s)$ through the abstraction function ϕ. In this case, the agent will no longer know precisely which state it is in, making the environment (a special case of) a POMDP [3, 6, 32, 52, 69, 70]. Abstraction can be seen as a special case of POMDPs because the observation results from *perceptual aliasing*, i.e., multiple states are perceived as the same. Perceptual aliasing may not be a problem when the resulting problem behaves as an MDP, as for a bisimulation [21, 45], but this is often not the case [1, 57, 70].

To formalize the combination of abstraction and RL in an MDP as a special case of a POMDP, we first give the general definition of an infinite horizon POMDP [34], which can be described by the tuple $\langle S, A, T, R, \Omega, O, \gamma \rangle$, where S, A, T, R, and γ are the same as in the MDP. The Ω is a finite set of observations $o \in \Omega$ that an agent can receive, and O is an observation function $O(o|a, s') = \Pr(o|a, s')$ that gives the probability of receiving an observation o after taking an action a and ending in state s'. Now, when an RL agent acts in an MDP but receives observations through ϕ, the uncertainty is only due to perceptual aliasing, which means that the observation is a deterministic function of the state: $O(o|a, s') = \Pr(o = \phi(s')|a, s') = \Pr(o = \phi(s')|s')$. For deterministic functions ϕ, this is 1 iff $o = \phi(s')$. The abstraction function ϕ has taken the role of the observation function O, with the observation space being $\bar{S}$.

Since we can view the combination of abstraction and RL in an MDP as a special case of a POMDP, RL methods for POMDPs could be used to find a solution. A common approach to finding solutions in POMDPs is through Bayesian RL, for which the Bayes-Adaptive POMDP (BA-POMDP) provides a framework [65]. Extensions of Bayesian RL for POMDPs are covered in the survey of Ghavamzadeh et al. [20]. In Deep RL, using recurrent neural networks is one way in which partial observability has been addressed [30, 78]. Specific focus has been on using variational inference methods [29, 75] and belief tracking [35, 47, 78]. However, these POMDP approaches are often general solutions for any POMDP, and they are not necessarily optimal for the special case of the POMDP induced by abstraction.

Instead of applying POMDP solution methods, it can be tempting to treat the resulting problem as a Markov problem and try to find a solution in this way. For instance, this could be tempting when the abstraction clusters together states with similar transition and reward functions in the abstract space, such as with a ϵ-bisimulation abstraction. However, it has been observed that treating this problem as a Markov process can lead to policies that are far from optimal,

and there could be no guarantee of finding an optimal solution [1, 57, 70]. In general, non-stationarity of the collected data, due to changing behavior of the policy, has been shown to lead to worse performance in Deep RL [28], and non-stationarity due to perceptual aliasing can lead to similar problems when not addressed. Therefore, to find good solutions, methods that combine RL and abstraction should take into account perceptual aliasing.

3 Utilizing Given Abstraction Functions

This section presents an overview of the literature that utilizes an abstraction function for MBRL. First, Section 3.1 discusses the relation between abstract MDPs and Robust MDPs (RMDPs) and how solution methods for RMDP can allow for obtaining better policies when using an abstract learned model. Section 3.2 considers the RL setting where we do not have such a model, but we are given some abstraction function ϕ and see how abstraction can be leveraged to improve performance. Section 3.3 deals with the setting where we are given a set of abstractions and have to learn which one leads to optimal performance. Afterward, Section 4 deals with the setting where we do not have an abstraction function ϕ and have to learn one online.

3.1 Robust Optimization

The RMDP [80], and the related Bounded Parameter MDP (BPMDP) [22], extend the MDP definition by allowing for uncertainty in the transition and reward functions, as quantified by intervals. This uncertainty is generally motivated by not having enough data to be sure about the transition functions but still being able to give some confidence intervals. Another motivation is inherent uncertainty, for instance caused by having a ϵ-bisimulation, where the uncertainty intervals are ϵ wide. If we learn an ϵ-bisimulation model and can estimate ϵ, we could apply solution methods for RMDP, this makes solution methods for RMDP interesting for RL with abstraction.

To solve problems with inherent uncertainty, the RMDP includes an additional set of outcomes B. The transition probabilities and reward function are a function of both $a \in A$ and $b \in B$. From a game-theoretic perspective, B can be seen as the actions of the adversary [61], which can be state-dependent and decide over the distribution of the transition and reward function from within the specified intervals. The solution to an RMDP also includes the policy of the adversary. For general uncertainty sets for B, it has been shown that the problem of finding an optimal robust policy is strongly NP-hard [80]. In order to find solutions in polynomial time, two main uncertainty sets for B have been considered: s-rectangular and s,a-rectangular sets [61, 80]. In the first case, the adversary can independently choose an outcome for each state s. In the second case, the adversary can choose an outcome for each state s and action a independently. Recent work has also given results when the uncertainty sets are less

strict [23, 51]. Taking into account the uncertainty with robust optimization can lead to better policies for the real environment [46, 80].

There has also been some work that combines abstraction with RMDP [46, 61]. The RAAM algorithm [61] receives an abstraction function and an MDP as input. It first constructs an RMDP and uses this to compute an approximately optimal policy for the original MDP. It is shown that this can be beneficial in the limit; bounds on the performance are given that are similar to the bounds for ϵ-bisimulation abstractions in planning [2]. The RAAM approach was later extended by Lim and Autef [46], who use a kernel-based approach, of which state abstraction can be seen as a special case.

The work in this section shows that uncertainty about the transition and reward functions can be dealt with in a principled way, given some uncertainty intervals. While some work connects this work to abstraction, it only focuses on results in the limit.

3.2 Leveraging an Abstraction Function

Often in RL, the environment will be unknown, but sometimes we have access abstraction function ϕ. This ϕ could, for instance, come from a domain expert or result from the discretization of a continuous problem. With ϕ, one could try to learn an abstract model, which is typically done by collecting data and then constructing a maximum likelihood model for the transition and reward functions in the abstract space. If we learn a correct abstract model and find the optimal policy for this problem, this solution can be near-optimal in the true MDP, depending on the abstraction used [2]. Learning in this way could be more sample-efficient than learning a model of the full MDP because the abstract space is smaller than the original state space.

One difficulty in this setting is learning a correct abstract model in the first place. In RL, samples can usually be considered independent, and this is used to show that an accurate model can be learned. In the combination of RL and abstraction, samples can no longer be considered independent due to perceptual aliasing [27, 69, 70]. In order to give sample efficiency results for RL plus abstraction, some work assumes that the collected samples are independent [17, 60]. The work by Paduraru et al. [60] assumes that they receive a data set with independent and identically distributed (i.i.d.) samples and show a trade-off between the quality of the abstraction and the quality of the transition model. The quality of the abstraction is measured in terms of the ϵ of ϵ-bisimulation. A larger ϵ means a coarser abstraction and a larger error. The second error relates to the number of samples we can get for a state-action pair, where a coarser abstraction gives more samples per state-action pair and a lower error. Like the work by Paduraru et al. [60], other work has also shown that the error of the agent can be decomposed into multiple components, which are based on the asymptotic bias of the representation and overfitting due to limited data (variance) [17, 67]. This bias-variance trade-off indicates that using abstractions can be especially beneficial when the available data is limited while being less beneficial when much data is available, which has been illustrated in experiments [17].

The assumption that the generated data consists of independent samples does not hold in general. Another way to show that we can learn an accurate abstract model is by looking at convergence in the limit. The convergence to an accurate estimation of the abstract model is possible under several conditions, e.g., when the policy is fixed or when the abstraction is a bisimulation [27, 69]. Having to use a fixed policy can be seen as a downside because a changing policy that explores helps to learn efficiently [71]. Another downside is that, in the limit, using the full model will be better than using an abstract model since only the error introduced by the bias remains, which is zero for the full model.

The work by Starre et al. [70] has recently shown that an accurate abstract model can still be learned by applying martingale theory [11]. They give the first finite-sample performance analysis for model-based RL plus abstraction by extending the results of an existing algorithm (R-MAX [9]) with the use of an ϵ-bisimulation abstraction.

This section shows that abstractions can lead to better performance with fewer data, trading it off with less accuracy when much data is available. For these methods to work, it is required to already have a good abstraction function, which can be challenging.

3.3 Abstraction Selection

While the work in the previous section mainly focused on the case where we have one particular abstraction function, there is also a considerable amount that has focused on state representation selection, where the agent is provided with a set of state representations (or abstraction functions). It is usually assumed that a domain expert provides these representations, and the goal is to select the best representation, often in terms of regret.

Most of this work focuses on finding representations that make the problem Markov instead of focusing on finding good approximate abstractions. In order to deal with perceptual aliasing, most work assumes that the provided set contains a Markov model of the environment [25, 48, 49, 54, 59]. In order to find a correct representation in the online setting, these algorithms eliminate non-Markov models by comparing the obtained rewards during execution with a threshold based on a Markov model. The work by Lattimore et al. [41] considers a similar setting where the dynamics of the true environment depend arbitrarily on the history of actions, rewards, and observations. Instead of getting a set of representation functions, they assume access to a given set of models, one of which is a correct model of the true environment. In this way, they can compare the calculated expected reward for the given model with the rewards obtained during the process and eliminate the unlikely models.

Other work does not assume that a Markov representation is available [31, 58], these both use an ϵ-bisimulation type abstraction. The work of Ortner et al. [58] builds on the work of Maillard et al. [48] by removing the necessity of having a Markov representation in the set of available representations. However, the analysis is invalid since there is an issue in the proof on which they build [18].

They also do not take into perceptual aliasing since they use a concentration inequality that requires i.i.d. samples. The work by Jiang et al. [31] deals with perceptual aliasing by explicitly assuming in their analysis that a data set consisting of samples that are i.i.d. is available. They give a performance bound for policies based on a learned abstract model and split the error into two components, similar to some of the work mentioned in Section 3.2 [17, 60]. These two components are used to create an algorithm that decides which representation should be used based on the available data.

The methods in this section show that we can learn to select a correct (Markov) representation, given an initial set of representations. Most of these methods are not very scalable, as they are tabular, and finding a good (Markov) representation/abstraction in larger problems can be challenging.

4 Online Abstraction Learning

The previously discussed works have mostly assumed that an abstract representation (or a set thereof) is readily available. However, this is not always possible. In this section, we consider the situation where such an abstraction is unavailable and has to be learned first while simultaneously learning about the environment. Two early studies on this topic provided promising experimental results [40, 52]. Section 4.1 covers tabular approaches, which have mostly been more theoretical, and Section 4.2 covers deep learned representations focused on scaling up.

4.1 Tabular Approaches

The combination of MBRL and abstraction has also been approached theoretically. The work by Bernstein and Shimkin [7] gives results for online abstraction when the transition functions are deterministic. The work by Ortner [57] explores the more general case of stochastic transition functions when trying to learn a ϵ-bisimulation. To learn a ϵ-bisimulation they maintain an interval on the estimation of the transition and reward functions for each state-action pair, which is used to create a BPMDP [22]. Subsequently, the BPMDP is abstracted by clustering the states that overlap in the transition and reward function for all actions, but only if they have a similar amount of samples. They give an example to show that clustered states must have a similar amount of samples for all the actions to obtain good performance. This is an interesting observation since it points out a problem that should be taken into account when learning an abstraction in combination with MBRL. A downside of the method is that it focuses on the computational benefit abstraction can bring; from the perspective of sample efficiency, a method that utilizes abstraction to learn more efficiently is desirable.

In the Bayesian RL setting, the work by Mandel et al. [50] proposes an algorithm that does online clustering and exploration. The clustering is done over state-action pairs rather than only over states. State-action abstractions allow for a broader class of abstractions since state abstractions can be considered a subset

of state-action abstractions while potentially still being optimality preserving. This gives additional power in doing the abstraction since, in some domains, there could be no similar states while similar state-action pairs exist. State-action pairs are grouped when the relative outcomes are likely to be the same. Relative outcomes are similar to observations. Given a relative outcome, the agent knows both the transition and reward. However, it needs to learn the distribution over relative outcomes for each state.

Work in block MDPs, or MDPs with rich observations, is a related approach where the assumption is that each state can generate multiple different observations [5, 15, 24, 39, 82]. Instead of having multiple states that generate the same observation (due to the abstraction function), each type of observation is only generated by one state, but each state can generate multiple observations. This is similar to representation learning, specifically to learning a bisimulation [5, 15, 82]. A common approach in this setting is to use spectral methods [5, 24, 39]. For these to work, it is necessary to be able to uniquely identify states from the observation function. While this is possible for model-irrelevance abstractions, this is generally not possible in the abstraction setting.

The focus of tabular approaches has been on block MDPs, which can lead to a considerable reduction in the state space in suitable problems. However, this does require the problem to have many states with the same behavior in an abstract space, i.e., there needs to be a bisimulation abstraction. This restricts the number of problems to which these methods can be applied.

4.2 Deep Learned Representations

There have also been several Deep RL approaches that focus on learning compact state representations, which can be viewed as an instance of state abstraction. For instance, the approaches by Sermanet et al. [68], Thomas et al. [74], Biza and Platt [8], François-Lavet et al. [16], Van der Pol et al. [63], Schrittwieser et al. [66], Allen et al. [3], and Ye et al. [81]. One crucial notion for abstraction in deep RL is a collapse of the latent representation [3, 12, 16, 63]. When considering only the transition function, it would be optimal to cluster all states into exactly one abstract state. It has been shown that losses that require both the transition and reward function of grouped states to be the same can avoid this collapse [19], making it essential to group states based on both transitions and rewards.

Recently, multiple contrastive methods have been used to learn compact representations for predicting the next state [4, 37, 56]. Their representation learning tries to maximize the mutual information between the present and future samples. To train the network, they use positive and negative next-state samples, where the positive samples are transitions that occurred, while the negative samples are transitions that did not occur. These negative samples should help prevent the potential collapse of the state representation. Their methods do not use the model to plan the policy but instead use actor-critic and policy optimization methods on top of the representation. The proposed representation learning method was able to help improve the performance of these methods.

Section	Method	Environment	Model	Abstraction ϕ	Abstraction Type	Theory	Scalability	Perceptual Aliasing
2.1	Planning	MDP	Given	Constructed	Many	V	$\sim$	Not an issue
	Tabular RL	Abstract MDP	Given	Build-in	Bisimulation related	V	X	Not an issue
3.1	Robust Optimization	MDP	Given (interval)	Build-in	Bisimulation	V	$\sim$	Assumption on uncertainty
	Robust Optimization	MDP + ϕ	Given	Given	Bisimulation related	$\sim$	X	Not an issue
3.2	Tabular RL	MDP + ϕ	Unknown	Given	Bisimulation related	V	X	Assumptions on data gathering
3.3	Abstraction Selection	MDP + $[\phi_1, \cdots, \phi_n]$	Unknown	Given	Several	V	X	Markov representation, assumption on data gathering
4.1	Tabular RL	MDP + ϕ	Unknown	Learned	Bisimulation related	V	X	Markov representation, specific check
	Bayesian RL	MDP + ϕ	Unknown	Learned	(s,a)-abstraction	V	X	Markov representation
	Spectral Methods	Block-MDP	Unknown	Learned	Bisimulation related	V	$\sim$	Markov representation
4.2	Deep RL	MDP + ϕ	Unknown	Learned	Bisimulation related	$\sim$	V	Markov representation, potential problem
	Contrastive Loss	MDP + ϕ	Unknown	Learned	Bisimulation related	X	V	Markov representation, potential problem
	Linear Latent Representations	MDP + ϕ	Unknown	Learned	Linear Function	X	V	Markov representation, potential problem

Table 1. Characterization of MBRL methods in combination with a type of state abstraction.

Other work has focused on learning deep representations for robotics [12, 33, 43]. This has investigated adding several types of robotic priors to bias the representation learning, which are added to the network as an auxiliary loss [30]. These priors encode knowledge about physics, e.g., that changes in the state are often gradual rather than abrupt. The state-representation objectives were instrumental in generalizing, as they significantly improved the results in the test domain. This shows that learning a compact model of the environment can be beneficial even if the model itself is not directly used for planning. Other methods for robotics focus on finding compact linear representations of a problem and finding a policy for this smaller model [76, 79, 83]. This has shown promising results for robotics, where many of the essential state features could be approximately linear.

Most of the work in this section focused on learning exact abstractions. They try to reduce the problem so that the resulting latent representation still makes the problem an MDP. This can be difficult to ensure, especially in Deep RL, so it is likely that the resulting representation is an approximate abstraction. Since most work does not acknowledge this, they do not consider the resulting perceptual aliasing, and algorithms can experience the problem illustrated by [57]: when states with a different number of visitations are grouped, this can lead to suboptimal policies. When this is not taken into account, this can lead an agent to be stuck in a suboptimal loop.

5 Discussion and Conclusion

We summarize our overview in Table 1, which compares the approaches on the type of environment, whether or not a model is given, how an abstraction ϕ is obtained, what kind of abstraction is used, available theoretical support, scalability, and how they deal with perceptual aliasing.

The methods in Sections 2 and 3 generally have strong theoretical support (V) in the form of bounded loss (e.g., [2, 80]), finite-sample guarantees (e.g., [60, 70]), or regret bounds (e.g., [49]). Most of these methods are not (X) scalable due to being tabular or only somewhat scalable ($\sim$) due to needing to be given a model, which in many cases is not possible. In most of these works, the problem of perceptual aliasing does not arise, either because of assumptions on data gathering or because an MDP, or MDP representation, is provided. Without assuming that samples are independent, Starre et al. [70] show that finite-sample bounds for MBRL in an MDP with an ϵ-bisimulation can be obtained. *Extending these results to other types of abstractions is still an open question.*

In Section 3.2, we saw a bias-variance trade-off with abstractions [17, 31, 60, 67]. Because of this trade-off, *an interesting direction would be to combine learning multiple representations with abstraction selection to decide which representation to use at which time.*

As discussed in Section 3.1, results for optimization under uncertainty could make it interesting to maintain confidence intervals for the learned models and use robust optimization to find policies. Since the model will generally not be

completely accurate during learning, robust optimization could improve performance [46]. Tabular work discussed in Sections 3.1 and 4.1 investigated this idea [57, 61], *scaling such approaches to larger problems is an interesting future direction.*

Most of the focus has been on abstractions related to bisimulation. As touched upon in Section 4.1, abstractions that aggregate state-action pairs can be more potent than state abstractions [50]. An open question is *what are the best types of abstraction to use*? Non-deterministic abstraction [69], temporal abstraction, or combinations of abstractions could be powerful but have not been as well studied [38].

In work by Schrittwieser et al. [66], there is some indication that, in online planning, using a coarser learned model rather than the true model can be beneficial. With limited planning time, planning with a compact learned model outperformed planning with the true model of the environment. *There could be a trade-off for learning between the coarseness of the model and the allotted planning time; a coarser model could perform better with a shorter planning time but worse with a longer planning time.*

The methods in Section 4.2 focus on learning abstractions that result in a Markov representation, e.g., bisimulation abstractions. However, *during learning, when the abstraction is likely not a Markov representation, perceptual aliasing occurs. How can the resulting non-stationarity be addressed?* In Section 4.1, we saw that the tabular work by Ortner [57] deals with perceptual aliasing, but to do so, it maintains visitation counts for all state-action pairs. Methods that can maintain counts in an approximate way, such as pseudo-counts [72], could enable a scalable version of the approach by Ortner [57]. Another approach to deal with perceptual aliasing in a more sample-efficient way could be using an algorithm such as ITER [28], which tackles the general non-stationarity of the data distribution caused by the RL algorithm. The idea of the algorithm is to frequently transfer the knowledge of the trained network to a new network and then use the new network for training. The knowledge is transferred through samples that are obtained from the collected data set as if they had been generated with the final policy of the trained network.

In multi-agent RL, the challenge is to behave optimally in the presence of other agents whose behavior may be non-stationary [26]. *Approaches for the multi-agent RL problem that address non-stationarity could be insightful for the combination of RL and abstraction.* One approach that could be relevant is trying to capture the non-stationarity that is the result of perceptual aliasing, which could, for instance, be done by using influence-based abstraction [55]. Influence-based abstraction aims to abstract a problem into a smaller local problem with a predictor that quantifies the influence of variables outside the local problem on the local problem. Given an accurate predictor, this results in a Markov problem. Such a predictor could capture the non-stationarity due to perceptual aliasing and improve performance. Influence-based abstraction has been applied together with Deep model-free RL, using a recurrent neural network to capture the influence, which has shown promising results [10].

Other approaches in multi-agent RL do not deal with the non-stationarity but simply ignore it by abstracting away the internal states of the other agents. Since this can be seen as a special case of the non-stationarity in the combination of RL and abstraction, insights from this combination on how to deal with non-stationarity as a result of perceptual aliasing could provide interesting directions for these multi-agent RL approaches.

Acknowledgements 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).

References

1. Abel, D., Arumugam, D., Lehnert, L., Littman, M.: State abstractions for lifelong reinforcement learning. In: ICML (2018)
2. Abel, D., Hershkowitz, D., Littman, M.: Near optimal behavior via approximate state abstraction. In: ICML (2016)
3. Allen, C., Parikh, N., Gottesman, O., Konidaris, G.: Learning markov state abstractions for deep reinforcement learning. In: NeurIPS (2021)
4. Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M.A., Hjelm, R.D.: Unsupervised state representation learning in atari. In: NeurIPS (2019)
5. Azizzadenesheli, K., Lazaric, A., Anandkumar, A.: Reinforcement learning in rich-observation mdps using spectral methods. arXiv (2016)
6. Bai, A., Srivastava, S., Russell, S.J.: Markovian state and action abstractions for mdps via hierarchical mcts. In: IJCAI (2016)
7. Bernstein, A., Shimkin, N.: Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains. ML (2010)
8. Biza, O., Platt, R.: Online abstraction with mdp homomorphisms for deep learning. arXiv (2018)
9. Brafman, R.I., Tennenholtz, M.: R-max-a general polynomial time algorithm for near-optimal reinforcement learning. JMLR (2002)
10. Suau de Castro, M., Congeduti, E., Starre, R., Czechowski, A., Oliehoek, F.: Influence-based abstraction in deep reinforcement learning. In: AAMAS Workshop on Adaptive Learning Agents (2019)
11. Chow, Y.S., Teicher, H.: Probability theory: independence, interchangeability, martingales. Springer Science & Business Media (2003)
12. De Bruin, T., Kober, J., Tuyls, K., Babuška, R.: Integrating state representation learning into deep reinforcement learning. RA-L (2018)
13. Dean, T., Givan, R.: Model minimization in markov decision processes. In: AAAI/IAAI (1997)

14. Dean, T., Givan, R., Leach, S.: Model reduction techniques for computing approximately optimal solutions for markov decision processes. In: UAI (1997)
15. Du, S., Krishnamurthy, A., Jiang, N., Agarwal, A., Dudik, M., Langford, J.: Provably efficient rl with rich observations via latent state decoding. In: ICML (2019)
16. François-Lavet, V., Bengio, Y., Precup, D., Pineau, J.: Combined reinforcement learning via abstract representations. In: AAAI (2019)
17. François-Lavet, V., Rabusseau, G., Pineau, J., Ernst, D., Fonteneau, R.: On overfitting and asymptotic bias in batch reinforcement learning with partial observability. JAIR (2019)
18. Fruit, R., Pirotta, M., Lazaric, A.: Near optimal exploration-exploitation in non-communicating markov decision processes. In: NeurIPS (2018)
19. Gelada, C., Kumar, S., Buckman, J., Nachum, O., Bellemare, M.G.: Deepmdp: Learning continuous latent space models for representation learning. In: ICML (2019)
20. Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A.: Bayesian reinforcement learning: A survey. Found. Trends Mach. Learn. (2015)
21. Givan, R., Dean, T., Greig, M.: Equivalence notions and model minimization in markov decision processes. Artif. Intell. (2003)
22. Givan, R., Leach, S., Dean, T.: Bounded-parameter markov decision processes. Artif. Intell. (2000)
23. Goyal, V., Grand-Clement, J.: Robust markov decision process: Beyond rectangularity. arXiv (2018)
24. Guo, Z.D., Doroudi, S., Brunskill, E.: A pac rl algorithm for episodic pomdps. In: AISTATS (2016)
25. Hallak, A., Di-Castro, D., Mannor, S.: Model selection in markovian processes. In: SIGKDD (2013)
26. Hernandez-Leal, P., Kaisers, M., Baarslag, T., de Cote, E.M.: A survey of learning in multiagent environments: Dealing with non-stationarity. arXiv (2017)
27. Hutter, M.: Extreme state aggregation beyond markov decision processes. TCS (2016)
28. Igl, M., Farquhar, G., Luketina, J., Boehmer, W., Whiteson, S.: Transient non-stationarity and generalisation in deep reinforcement learning. In: ICLR (2021)
29. Igl, M., Zintgraf, L., Le, T.A., Wood, F., Whiteson, S.: Deep variational reinforcement learning for pomdps. In: ICML (2018)
30. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv (2016)
31. Jiang, N., Kulesza, A., Singh, S.: Abstraction selection in model-based reinforcement learning. In: ICML (2015)
32. Jie, N.: Representation learning for model-based reinforcement learning: A survey. tinyurl.com/jieRep (2021)
33. Jonschkowski, R., Brock, O.: Learning state representations with robotic priors. Auton. Robots (2015)
34. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artif. Intell. (1998)
35. Karkus, P., Hsu, D., Lee, W.S.: Qmdp-net: Deep learning for planning under partial observability. arXiv (2017)
36. Keith, A.J., Ahner, D.K.: A survey of decision making and optimization under uncertainty. Ann. Oper. Res. (2021)
37. Kipf, T., van der Pol, E., Welling, M.: Contrastive learning of structured world models. In: ICLR (2019)

38. Konidaris, G.: On the necessity of abstraction. Curr Opin Behav Sci (2019)
39. Krishnamurthy, A., Agarwal, A., Langford, J.: Pac reinforcement learning with rich observations. arXiv (2016)
40. Kuvayev, L., Sutton, R.S.: Model-based reinforcement learning with an approximate, learned model. In: Yale Workshop on Adaptive and Learning Systems (1996)
41. Lattimore, T., Hutter, M., Sunehag, P.: The sample-complexity of general reinforcement learning. In: ICML (2013)
42. Lesort, T., Díaz-Rodríguez, N., Goudou, J.F., Filliat, D.: State representation learning for control: An overview. Neural Netw. (2018)
43. Lesort, T., Seurin, M., Li, X., Díaz-Rodríguez, N., Filliat, D.: Deep unsupervised state representation learning with robotic priors: a robustness analysis. In: IJCNN (2019)
44. Li, L., Walsh, T.J., Littman, M.L.: Towards a unified theory of state abstraction for mdps. In: ISAIM (2006)
45. Li, L.: A unifying framework for computational reinforcement learning theory. Ph.D. thesis, Rutgers University-Graduate School-New Brunswick (2009)
46. Lim, S.H., Autef, A.: Kernel-based reinforcement learning in robust markov decision processes. In: ICML (2019)
47. Ma, X., Karkus, P., Hsu, D., Lee, W.S., Ye, N.: Discriminative particle filter reinforcement learning for complex partial observations. In: ICLR (2019)
48. Maillard, O.A., Nguyen, P., Ortner, R., Ryabko, D.: Optimal regret bounds for selecting the state representation in reinforcement learning. In: ICML (2013)
49. Maillard, O.A., Ryabko, D., Munos, R.: Selecting the state-representation in reinforcement learning. In: NeurIPS (2011)
50. Mandel, T., Liu, Y.E., Brunskill, E., Popovic, Z.: Efficient bayesian clustering for reinforcement learning. In: IJCAI (2016)
51. Mannor, S., Mebel, O., Xu, H.: Robust mdps with k-rectangular uncertainty. MOOR (2016)
52. McCallum, A.K.: Reinforcement learning with selective perception and hidden state. U of Rochester (1996)
53. Moerland, T.M., Broekens, J., Jonker, C.M.: Model-based reinforcement learning: A survey. arXiv (2020)
54. Nguyen, P., Maillard, O.A., Ryabko, D., Ortner, R.: Competing with an infinite set of models in reinforcement learning. In: AISTATS (2013)
55. Oliehoek, F.A., Witwicki, S.J., Kaelbling, L.P.: Influence-based abstraction for multiagent systems. In: AAAI (2012)
56. Van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv (2018)
57. Ortner, R.: Adaptive aggregation for reinforcement learning in average reward markov decision processes. Ann. Oper. Res. (2013)
58. Ortner, R., Maillard, O.A., Ryabko, D.: Selecting near-optimal approximate state representations in reinforcement learning. In: ALT (2014)
59. Ortner, R., Pirotta, M., Lazaric, A., Fruit, R., Maillard, O.A.: Regret bounds for learning state representations in reinforcement learning. In: NeurIPS (2019)
60. Paduraru, C., Kaplow, R., Precup, D., Pineau, J.: Model-based reinforcement learning with state aggregation. In: European Workshop on RL (2008)
61. Petrik, M., Subramanian, D.: Raam: The benefits of robustness in approximating aggregated mdps in reinforcement learning. In: NeurIPS (2014)
62. Plaat, A., Kosters, W., Preuss, M.: High-accuracy model-based reinforcement learning, a survey. arXiv (2021)

63. Van der Pol, E., Kipf, T., Oliehoek, F.A., Welling, M.: Plannable approximations to mdp homomorphisms: Equivariance under actions. In: AAMAS (2020)
64. Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley (2014)
65. Ross, S., Pineau, J., Chaib-draa, B., Kreitmann, P.: A bayesian approach for learning and planning in partially observable markov decision processes. JMLR (2011)
66. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature (2020)
67. Serban, I.V., Sankar, C., Pieper, M., Pineau, J., Bengio, Y.: The bottleneck simulator: A model-based deep reinforcement learning approach. JAIR (2020)
68. Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., Brain, G.: Time-contrastive networks: Self-supervised learning from video. In: ICRA (2018)
69. Singh, S.P., Jaakkola, T., Jordan, M.I.: Reinforcement learning with soft state aggregation. In: NeurIPS (1995)
70. Starre, R.A.N., Loog, M., Oliehoek, F.A.: An analysis of abstracted model-based reinforcement learning. arXiv (2022)
71. Strehl, A.L., Littman, M.L.: An analysis of model-based interval estimation for markov decision processes. JCSS (2008)
72. Taïga, A.A., Courville, A., Bellemare, M.G.: Approximate exploration through state abstraction. arXiv (2018)
73. Talvitie, E.: Model regularization for stable sample rollouts. In: UAI (2014)
74. Thomas, V., Bengio, E., Fedus, W., Pondard, J., Beaudoin, P., Larochelle, H., Pineau, J., Precup, D., Bengio, Y.: Disentangling the independently controllable factors of variation by interacting with the world. arXiv (2018)
75. Tschiatschek, S., Arulkumaran, K., Stühmer, J., Hofmann, K.: Variational inference for data-efficient model learning in pomdps. arXiv (2018)
76. Van Hoof, H., Chen, N., Karl, M., van der Smagt, P., Peters, J.: Stable reinforcement learning with autoencoders for tactile and visual data. In: IROS (2016)
77. Wald, A.: Statistical Decision Functions. John Wiley (1950)
78. Wang, Y., Tan, X.: Deep recurrent belief propagation network for pomdps. In: AAAI (2021)
79. Watter, M., Springenberg, J.T., Boedecker, J., Riedmiller, M.: Embed to control: a locally linear latent dynamics model for control from raw images. In: NeurIPS (2015)
80. Wiesemann, W., Kuhn, D., Rustem, B.: Robust markov decision processes. MOOR (2013)
81. Ye, W., Liu, S., Kurutach, T., Abbeel, P., Gao, Y.: Mastering atari games with limited data. In: NeurIPS (2021)
82. Zhang, A., Lyle, C., Sodhani, S., Filos, A., Kwiatkowska, M., Pineau, J., Gal, Y., Precup, D.: Invariant causal prediction for block mdps. In: ICML (2020)
83. Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., Levine, S.: Solar: Deep structured representations for model-based reinforcement learning. In: ICML (2019)