The Value of Abstraction

Mark K. Ho
Princeton University

David Abel
Brown University

Thomas L. Griffiths
Princeton University

Michael L. Littman
Brown University

Agents that can make better use of computation, experience, time, and memory can solve a greater range of problems more effectively. A crucial ingredient for managing such finite resources is intelligently chosen *abstract representations*. But, how do abstractions facilitate problem solving under limited resources? What makes an abstraction useful? To answer such questions, we review several trends in recent reinforcement-learning research that provide insight into how abstractions interact with learning and decision making. During learning, abstraction can guide exploration and generalization as well as facilitate efficient tradeoffs—e.g., time spent learning versus the quality of a solution. During computation, good abstractions provide simplified models for computation while also preserving relevant information about decision-theoretic quantities. These features of abstraction are not only key for scaling up artificial problem solving, but can also shed light on what pressures shape the use of abstract representations in humans and other organisms.

Keywords: abstraction, reinforcement learning, bounded rationality, planning, problem solving, rational analysis

Highlights

- Agents with finite space, time, and data can benefit from abstract representations.
- Both humans and artificial agents rely on abstractions to solve complex problems.
- Recent work in reinforcement learning elucidates what makes abstractions useful.
- Abstractions guide learning, facilitate tradeoffs, and simplify computation.
- This structure provides a basis for a rational analysis of abstract representations.

Pre-print June 2019—to appear in Current Opinion in Behavioral Sciences themed issue on Artificial Intelligence.

Edited by Matt Botvinick and Sam Gershman.

We would like to thank the anonymous reviewers for their comments that greatly improved previous versions of this paper. This material is based upon work supported by the NSF under Grant No. 1544924.

Correspondence concerning this article should be addressed to Michael L. Littman, Brown University, Providence, RI. Email: mlittman@cs.brown.edu.

Introduction

In spite of bounds on space, time, and data, people are able to make good decisions in complex scenarios. What enables us to do so? And what might equip artificial systems to do the same? One essential ingredient for making complex problem solving tractable is a capacity for exploiting *abstract representation*.

To illustrate why abstractions are needed for adaptive decision making, consider hiking in a forest (Figure 1). A river runs from mountains in the north down south through the forest, and you have set up camp in a clearing just east of the river. As the sun begins to set, you find yourself west of the river and want to return to camp. How do you accomplish this task? You might reason, "Since I am on the *west side of the river*, I probably want to *cross the river* to get to the *east side of the river* and then *walk towards my campsite*." Such a compact thought is possible because you have abstract representations for that support decision making: These include the general notion of being in a location relative to the axis of the river, or the rough ideas of crossing the river and moving towards the campsite.

Without abstractions, decision making would be confined to specific, low-level states and actions—e.g., "At the current state, rotate your left leg 25°, place it down beside the rock on the path, then swing your arm forward...". In a huge and complex scenario like navigating through a forest, this approach is both cognitively unwieldy and, in a fundamental

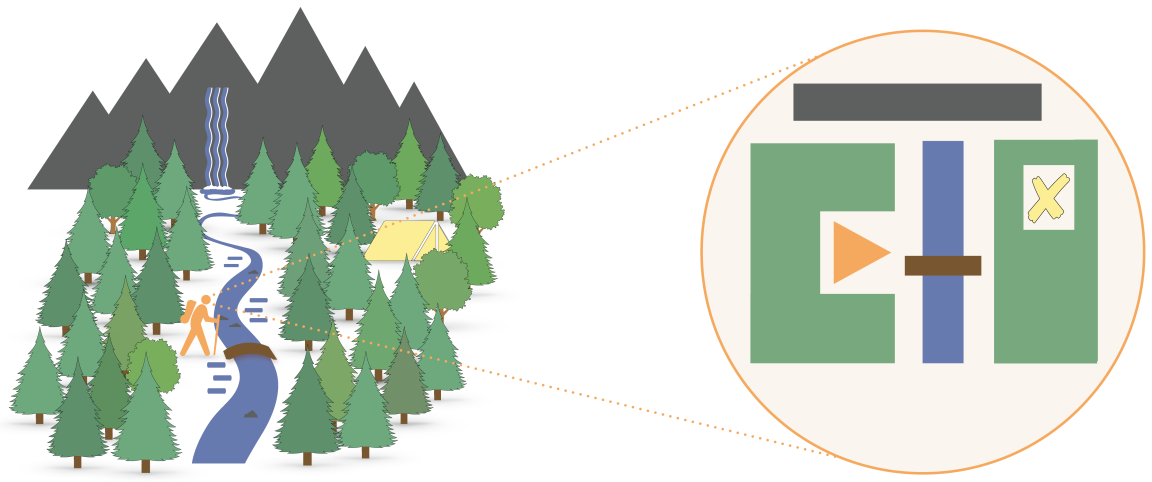

Figure 1. An environment, an agent, and the agent's abstract representation of a domain.

sense, computationally intractable (Bellman, 1957; Bylander, 1994). The difficulty is due to decision-making costs that arise from two interrelated sources. First, agents need to learn about their environment, which costs time and experience (e.g., via exploration) (Kakade, 2003). Second, agents need to compute decision-theoretic quantities, which costs memory and thought (e.g., via planning) (Littman, Dean, & Kaelbling, 1995). As problems grow, these costs grow exponentially. Abstractions are essential because they enable decision makers to manage the growth of these costs.

Within psychology and neuroscience, abstraction plays a key role in the hierarchical learning and organization of motor sequences (Lashley, 1951; Rosenbaum, Kenny, & Derr, 1983; Rosenbaum, Inhoff, & Gordon, 1984), habits (Botvinick, 2008; Dezfouli & Balleine, 2013; Ribas-Fernandes et al., 2011), and planning (Solway et al., 2014; Botvinick & Weinstein, 2014). For this reason, researchers in artificial intelligence (AI) and reinforcement learning (RL) have long been interested in how to leverage abstraction to provide human-like solutions to human-level problems (Dayan & Hinton, 1993; Sacerdoti, 1974; Dietterich, 2000; Givan, Dean, & Greig, 2003; Andre & Russell, 2002; Barto & Mahadevan, 2003). In this paper, we discuss work in AI and RL that helps elucidate how abstractions support efficient decision making in both people and machines.

Research in AI and RL focuses on two broad types of abstraction. First, *state abstractions* treat certain configurations of the environment as similar by aggregating them or assigning them shared features. For instance, in the hiking example, *being west of the river* is not a single concrete configuration of the environment, but rather an abstract representation that covers many concrete states (e.g., being different distances west of the river). Second, *temporal abstractions* (often called "options") are temporally extended macro-actions that describe a general course of action. For example, the idea of *crossing the bridge* is not a single specific action, but rather captures many possible courses of action.

State and temporal abstractions enable compact representation of a domain. For instance, Figure 2a represents the hiking example as a graph where nodes are specific configurations of the environment (i.e., ground states) and edges are transitions to new configurations resulting from taking actions. The appropriate abstractions could induce the simpler problem representation in Figure 2b. This new model ignores irrelevant details like the color of specific trees while capturing useful distinctions such as your location relative to the river. As a result, it supports efficient learning and requires little thought to reason about compared to the original version.

The foregoing discussion sketches out how abstractions can support efficient decision making. But how could it work in practice? Here, we dive into recent work in AI and RL that provides some answers. Specifically, we focus on three ways in which abstractions have been shown to facilitate efficient and scalable decision making. First, abstractions *guide exploration and generalization* by systematically modifying the distribution of learning experiences. In particular, abstractions can guide how agents explore and generalize based on environmental structure or representational simplicity. Second, abstractions *facilitate efficient tradeoffs in learning*. For example, they enable learning time and optimality to be exchanged as well as support optimal transfer between tasks. Finally, abstraction is essential for *simplifying computation*. For instance, abstractions influence the cost of computing a good plan by inducing simpler or more complex planning models. In the following sections, we discuss each of these benefits of abstraction.

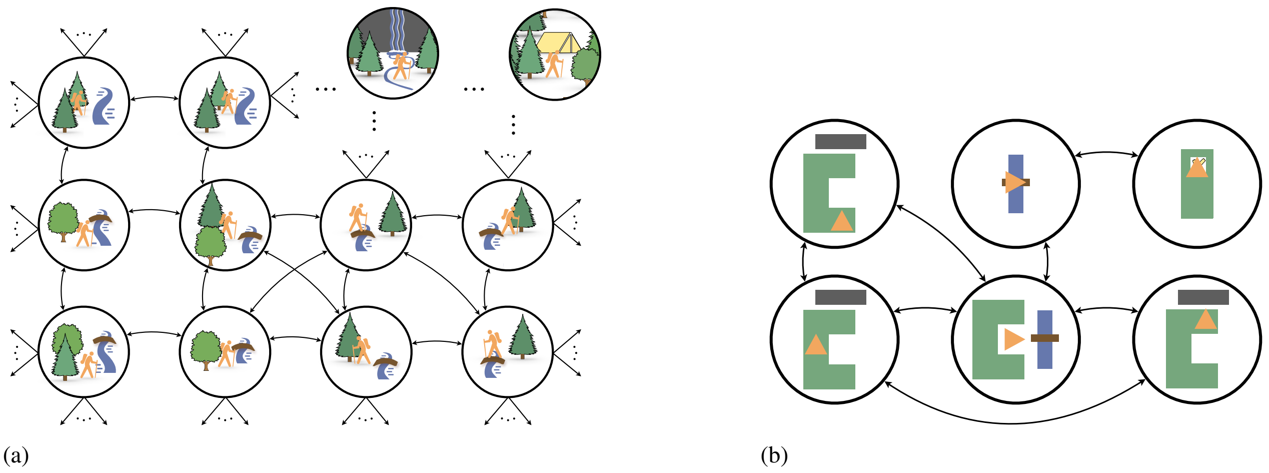
(a) (b)

Figure 2. (a) A domain in all of its complexity, and (b) the representation induced by an effective set of state and temporal abstractions for this domain. The representation in (b) facilitates efficient learning and computation, unlike the one in (a).

Abstractions Guide Exploration and Generalization

Suppose during your camping trip you want to learn to fish. At first, you try a few spots along the river at random. But soon, you notice a pattern: Certain areas have more vegetation, others have less. This information provides a new and more efficient way for you to organize your fishing attempts. For example, you could fish areas with high and low vegetation to gain a range of experiences about how it affects your catch. Or, on your first trip you may learn that your best catch was in high vegetation areas, so that on your second trip to a different river, you seek out similar areas. Here, the abstract concept of river vegetation guides your exploration in the current fishing task and tracks a generalizable feature relevant to future tasks, which both allow you to make better use of your limited time and experience.

In RL, abstractions similarly facilitate efficient learning by guiding exploration and generalization. But what is the basis of this guidance? Put another way, the concept of vegetation is useful, but what determines the identification of such a concept in general? Below, we discuss two ways in which abstractions can guide learning: domain structure and representational simplicity.

Domain Structure

Many domains have meaningful structure based on the topology of state connections. In the hiking example, *being west of the river* is an intuitive abstract state because all the locations west of the river are more connected to one another and less connected to locations east of the river. Early research in RL identified a number strategies for learning abstractions based on clustering or graph-theoretic notions like “bottleneck” states (Şimşek & Barto, 2004). More recent methods generalize such approaches to learn abstractions based on the state connectivity induced by a domain’s structure. For instance, the *successor representation* (Dayan, 1993; Gershman, Moore, Todd, Norman, & Sederberg, 2012; Momennejad et al., 2017) and *successor features* (Barreto et al., 2017; Kulkarni, Saeedi, Gautam, & Gershman, 2016) are state abstractions based on how likely an agent is to visit other states or features. A similar idea underpins *eigen-options* (Machado, Bellemare, & Bowling, 2017; Machado, Rosenbaum, et al., 2017; Mahadevan, 2005). Specifically, both successor representations and eigen-options involve decomposing an a domain’s structure into a set of “principal components” that compactly represent how certain regions of a domain are connected or will be visited.

Thus, *being west of the river* and *being east of the river* could emerge as successor-based state abstractions due to the connectivity and visitation of their respective ground states. A corresponding eigen-option would express the temporally abstract action of *going from the west side to the east side* since action sequences falling in this category capture a principal source of variance in connectivity. Notably, abstractions based on state connectivity encode information about the dynamics of a domain separate from its reward structure. These reward-agnostic representations can be learned either offline or during task learning, but, in either case, they provide an especially powerful and efficient way to adapt when rewards change or when transferring to tasks with related dynamics.

Representational Simplicity

Abstractions also guide exploration and generalization based on representational simplicity. Simplicity plays a key role in understanding representations in humans (Chater & Vitányi, 2003; Sanborn, Bourgin, Chang, & Griffiths, 2018) and machines (Li & Vitányi, 2008). For finite decision makers, simplicity is important because it enables *compression*, the representation of information using less physical space (e.g., memory).

In RL, a number of methods have been developed that leverage a bias towards representational simplicity. For example, *curiosity-driven learning* (Schmidhuber, 1991; Oudeyer, Kaplan, & Hafner, 2007; Bellemare et al., 2016; Pathak, Agrawal, Efros, & Darrell, 2017; Ostrovski, Bellemare, Oord, & Munos, 2017; Kulkarni, Narasimhan, Saeedi, & Tenenbaum, 2016) pits a motivation to learn a simple, compressed representation of the world against a desire for new experiences that do not easily fit within that representation. Placing these two processes in competition with one another prevents the drive for simplicity from compressing everything into a single undifferentiated representation (the ultimate abstraction) that cannot support high quality decisions and instead pushes the system to adaptively explore a range of qualitatively different parts of a task. Doing so ultimately allows the architecture to both represent the world compactly and explore it efficiently.

Simplicity also plays a role in the Option-Critic framework (Bacon, Harb, & Precup, 2017; Harb, Bacon, Klissarov, & Precup, 2018; Liu, Machado, Tesauro, & Campbell, 2017), a neural network architecture that simultaneously learns abstract macro-actions and the value of low-level actions that constitute them. By concurrently learning at multiple levels of abstraction, the agent acquires a representation that leads to better transfer to new tasks. The process at play is closely related to the notion of a "blessing of abstraction" in causal learning (Goodman, Ullman, & Tenenbaum, 2011) and regularization in supervised learning (Bishop, 2006): A bias towards simple, unifying representations not only saves on representational space, but also prevents overfitting to a particular task. An important consequence of this bias is that the agent is able to transfer to new tasks more efficiently.

Abstractions Facilitate Efficient Tradeoffs

Decision making and learning involve tradeoffs when space, time, and data are limited. But, it is important that these tradeoffs be made *efficiently* (i.e., without unnecessary costs). Here, we discuss how abstractions can efficiently trade off learning time and optimality, as well as minimize negative transfer between tasks.

Learning Time and Optimality

Imagine that every morning you wake up at your campsite and need to navigate to the bridge crossing the river. What kind of abstract states would be useful? It depends on how much time you have to spend learning. One option is to learn every tree, rock, and shortcut through the forest in great detail. Doing so has the benefit of ensuring you learn a highly efficient route to the bridge. An alternative strategy is to learn a less detailed representation of the forest that takes you to an obvious landmark—e.g., the river—and then take a simple path to the bridge. This second approach may be more costly in terms of the physical actions taken or time navigating, but is simpler and faster to learn. Depending on how much time and data are available, it may be better for the agent to pursue the second strategy than the first.

Learning the simpler but suboptimal policy depends on the resolution of the state abstractions used—i.e., the amount of detail one is willing to keep or ignore about the environment. In RL, a number of approaches use abstraction in this manner to efficiently balance optimality and learning time (Jiang, Kulesza, & Singh, 2015; Ortner, 2013). For instance, if abstractions are represented by aggregating ground states into clusters, adaptively modulating the size of the clusters and granularity of the state representation controls this tradeoff (Taiga, Courville, & Bellemare, 2018; Bellemare et al., 2016; Mandel, Liu, Brunskill, & Popovic, 2016).

Optimizing Transfer between Tasks

Abstractions also facilitate effective transfer from one task to another. Transfer can be positive (Sutton, Precup, & Singh, 1999; Konidaris & Barto, 2007; Bacon et al., 2017; Topin et al., 2015), but it can also be negative (Jong, Hester, & Stone, 2008). Ideally, an agent would learn abstractions that avoid negative transfer as much as possible while increasing the possibility of positive transfer. One setting in which this approach is feasible is in *lifelong learning*, in which an agent is assumed to be continuously receiving tasks from a distribution of tasks. Given separate phases of discovering and using temporal abstractions, it is possible to ensure that any learned options only help later learning and decision making (Brunskill & Li, 2014). Used properly, temporal abstractions can also guarantee that the amount of data needed for learning is reduced even within a single task (Fruit & Lazaric, 2017; Fruit, Pirotta, Lazaric, & Brunskill, 2017). For agents with limited time, such abstractions can play a critical role in ensuring efficient and effective use of data.

Abstractions Simplify Computation

Computation is costly for people (Griffiths, Lieder, & Goodman, 2015) as well as artificial agents (Russell, 1997). For example, if you were to plan a hike back to your campsite, you could simulate possible routes from your current state or "work backwards" from your final destination. These cognitive operations require time and mental effort. However, the total cost of computation depends directly on the model such operations occur over (Givan et al., 2003). Planning can be easier given a good abstract representation (e.g., compare Figure 2a and Figure 2b). Recent work in RL leverages this intuition to efficiently scale planning to complex domains. Here, we focus on two broad approaches: *algorithm-specific abstractions*, in which an abstraction scheme is tailored to take advantage of particular features of a planning algorithm, and *end-to-end* approaches, in which learning abstractions for planning occurs in the context of a more general learning architecture.

Algorithm-Specific Abstractions

Algorithms such as Monte Carlo Tree Search (Abramson, 1987; Coulom, 2006; Browne et al., 2012) are based on the notion of *forward simulation*: From an initial state, candidate plans are generated, evaluated, and the best is chosen. The major problem for these algorithms is branching: Each additional timestep into the future increases the number of plans to evaluate multiplicatively. To keep branching manageable, state-aggregation techniques simplify models by only storing relevant information. For instance, by treating all states with the same optimal action as identical, branching due to stochasticity in action outcomes can be reduced (Hostetler, Fern, & Dietterich, 2014; Anand, Grover, Mausam, & Singla, 2015; Anand, Noothigattu, Mausam, & Singla, 2016). Along similar lines, the model used for planning can be simplified by aggregating states that lead to the same possible outcomes within a few time steps (Jiang, Singh, & Lewis, 2014).

An alternative to forward simulation is *backwards induction*, which involves starting from possible future states and propagating information backwards to possible precursor states. Abstractions can also assist computation in backward induction. For instance, value iteration (Bellman, 1957) repeatedly "sweeps" over a set of states and backs up information to each precursor state. Since each iteration involves looking at every state in the planning model, it is extremely sensitive to the number of possible states. However, by incorporating temporal abstractions into the planning model, more information can be propagated further backwards on each sweep (Mann & Mannor, 2014) and it becomes possible to more efficiently trade off error in the approximation of future rewards with computation (Mann, Mannor, & Precup, 2015).

Finally, relaxing the goal of planning provides opportunities to leverage powerful abstractions. For instance, *plan feasibility* seeks to determine whether a plan can be constructed that satisfies some set of constraints. Since the goal is no longer *optimality* but simply *satisfiability*, relevant aspects of the task can be translated into symbolic logic and efficiently tested (Konidaris, Kaelbling, & Lozano-Perez, 2014, 2018; McDermott et al., 1998).

End-to-end Efficient Planning

Another approach is to learn planning abstractions while also learning everything else about a task in an *end-to-end* manner with neural networks. Using abstractions for planning is especially important because planning in a ground state space (e.g., the pixels of a video game) is only possible for short time horizons (Oh, Guo, Lee, Lewis, & Singh, 2015). To address these limitations, architectures have been developed that learn and use abstract representations for planning. In particular, the abstract representation needs to be trained to predict future rewards, thus preserving the most relevant information for decision making in the planning model (Oh, Singh, & Lee, 2017; Silver et al., 2017).

Conclusion

Abstractions are key for efficiently scaling decision making. In particular, recent research in RL demonstrates how state and temporal abstractions guide exploration and generalization, facilitate efficient tradeoffs, and simplify computations. In the context of learning and decision making with limited resources, abstractions enable agents to strategically manage finite space, time, and data, which is necessary for scaling problem solving to complex tasks.

For psychologists, this analysis raises questions about why biological agents like humans have certain types of abstract representations. For instance, humans and other organisms may have certain abstractions not only because they facilitate effective exploration and generalization but because they support fast planning or modulate critical tradeoffs during learning. Future work in both computer science and psychology will need to identify other pressures that can shape abstraction learning—such as the need to communicate and coordinate with others—to offer a clearer understanding of the value of abstraction.

Conflict of Interest Statement

The authors declare no conflict of interest.

Annotated References

- (**) (Bacon et al., 2017) - Proposes *option-critic* architecture that demonstrates the effectiveness of jointly learning abstract and low-level policies.

- (*) (Jiang et al., 2015) - Analyzes how state aggregation learning schemes can be used to efficiently trade off data and solution quality, which is key when data is costly or finite.

- (*) (Machado, Rosenbaum, et al., 2017) - Combines learning eigen-options with learning successor representations, which both leverage the transition structure of a domain for learning abstract representations.

- (**) (Pathak et al., 2017) - Implements curiosity as an intrinsic reward signal based on an agent's error when predicting the outcome of actions to enable effective representation learning even in the absence of external rewards.

- (*) (Oh et al., 2017) - Proposes the *Value Prediction Network* architecture that learns an internal model composed of abstract states for efficient model-based planning.

- (*) (Barreto et al., 2017) - Introduces a generalization of successor representations (Dayan, 1993) that enables efficient transfer across tasks with shared causal structure.

References

Abramson, B. D. (1987). *The Expected-Outcome Model of Two-Player Games* (Unpublished doctoral dissertation). Columbia University, New York, NY, USA.

Anand, A., Grover, A., Mausam, & Singla, P. (2015). ASAP-UCT: Abstraction of State-Action Pairs in UCT. *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Anand, A., Noothigattu, R., Mausam, & Singla, P. (2016). OGA-UCT: On-the-Go Abstractions in UCT. *Twenty-Sixth International Conference on Automated Planning and Scheduling*.

Andre, D., & Russell, S. J. (2002). State Abstraction for Programmable Reinforcement Learning Agents. In *Eighteenth national conference on artificial intelligence* (pp. 119–125). Menlo Park, CA, USA: American Association for Artificial Intelligence.

Bacon, P.-L., Harb, J., & Precup, D. (2017). The option-critic architecture. In *Proceedings of the conference aaai* (pp. 1726–1734).

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., & Silver, D. (2017). Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems* (pp. 4055–4065).

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, *13*(1-2), 41–77.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems* (pp. 1471–1479).

Bellman, R. (1957). *Dynamic programming*. Princeton University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Botvinick, M. M. (2008). Hierarchical models of behavior and prefrontal function. *Trends in Cognitive Sciences*, *12*(5), 201–208. doi: 10.1016/J.TICS.2008.02.009

Botvinick, M. M., & Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *369*(1655), 20130480–20130480. doi: 10.1098/rstb.2013.0480

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., ... Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, *4*(1), 1–43.

Brunskill, E., & Li, L. (2014). PAC-inspired option discovery in lifelong reinforcement learning. In *International conference on machine learning* (pp. 316–324).

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *69*, 161–204.

Chater, N., & Vitányi, P. (2003). Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences*, *7*(1), 19–22. doi: 10.1016/S1364-6613(02)00005-0

Coulom, R. (2006). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *International conference on computers and games* (pp. 72–83).

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, *5*(4), 613–624.

Dayan, P., & Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems* (pp. 271–278).

Dezfouli, A., & Balleine, B. W. (2013). Actions, Action Sequences and Habits: Evidence That Goal-Directed and Habitual Action Control Are Hierarchically Organized. *PLoS Computational Biology*, *9*(12), e1003364. doi: 10.1371/journal.pcbi.1003364

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Fruit, R., & Lazaric, A. (2017). Exploration–exploitation in MDPs with options. *arXiv preprint arXiv:1703.08667*.

Fruit, R., Pirotta, M., Lazaric, A., & Brunskill, E. (2017). Regret minimization in MDPs with options without prior knowledge. In *Advances in Neural Information Processing Systems* (pp. 3169–3179).

Gershman, S. J., Moore, C. D., Todd, M. T., Norman, K. A., & Sederberg, P. B. (2012). The successor representation and temporal context. *Neural Computation*, *24*(6), 1553–1568.

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, *147*(1-2), 163–223.

Goodman, N. D., Ullman, T. D., & Tenenbaum, J. B. (2011). Learning a theory of causality. *Psychological review*, *118*(1), 110.

Griffiths, T. L., Lieder, F., & Goodman, N. D. (2015). Rational Use of Cognitive Resources: Levels of Analysis Between the Computational and the Algorithmic. *Topics in Cognitive Science*, *7*(2), 217–229. doi: 10.1111/tops.12142

Harb, J., Bacon, P.-L., Klissarov, M., & Precup, D. (2018). When Waiting Is Not an Option: Learning Options With a Deliberation Cost. *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hostetler, J., Fern, A., & Dietterich, T. (2014). State Aggregation in Monte Carlo Tree Search. *AAAI 2014*, 7.

Jiang, N., Kulesza, A., & Singh, S. (2015). Abstraction selection in model-based reinforcement learning. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning* (Vol. 37, pp. 179–188). Lille, France: PMLR.

Jiang, N., Singh, S., & Lewis, R. (2014). Improving UCT planning via approximate homomorphisms. In *Proceedings of the 2014 international conference on autonomous agents and multi-agent systems* (pp. 1289–1296).

Jong, N. K., Hester, T., & Stone, P. (2008). The utility of temporal abstraction in reinforcement learning. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems* (pp. 299–306).

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning* (Unpublished doctoral dissertation). University of London, England.

Konidaris, G., & Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence* (Vol. 7, pp. 895–900).

Konidaris, G., Kaelbling, L. P., & Lozano-Perez, T. (2014). Constructing symbolic representations for high-level planning. In *Aaai* (pp. 1932–1938).

Konidaris, G., Kaelbling, L. P., & Lozano-Perez, T. (2018). From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence Research*, *61*, 215–289.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems* (pp. 3675–3683).

Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Lashley, K. S. (1951). The problem of serial order in behavior. In *Cerebral mechanisms in behavior; the hixon symposium.* (pp. 112–146). Oxford, England: Wiley.

Li, M., & Vitányi, P. (2008). *An Introduction to Kolmogorov Complexity and Its Applications*. New York, NY: Springer New York. doi: 10.1007/978-0-387-49820-1

Littman, M. L., Dean, T. L., & Kaelbling, L. P. (1995). On the complexity of solving markov decision problems. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence* (pp. 394–402).

Liu, M., Machado, M. C., Tesauro, G., & Campbell, M. (2017). The eigenoption-critic framework. *arXiv preprint arXiv:1712.04065*.

Machado, M. C., Bellemare, M. G., & Bowling, M. (2017). A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956*.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., & Campbell, M. (2017). Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on machine learning* (pp. 553–560).

Mandel, T., Liu, Y.-E., Brunskill, E., & Popovic, Z. (2016). Efficient bayesian clustering for reinforcement learning. *Proceedings of the International Joint Conference on Artificial Intelligence*.

Mann, T., & Mannor, S. (2014). Scaling up approximate value iteration with options: Better policies with fewer iterations. In *International conference on machine learning* (pp. 127–135).

Mann, T., Mannor, S., & Precup, D. (2015). Approximate value iteration with temporally extended actions. *Journal of Artificial Intelligence Research*, *53*, 375–438.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., … Wilkins, D. (1998). Pddl-the planning domain definition language.

Momennejad, I., Russek, E. M., Cheong, J. H., Botvinick, M. M., Daw, N. D., & Gershman, S. J. (2017). The successor representation in human reinforcement learning. *Nature Human Behaviour*, *1*(9), 680–692. doi: 10.1038/s41562-017-0180-8

Oh, J., Guo, X., Lee, H., Lewis, R. L., & Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems* (pp. 2863–2871).

Oh, J., Singh, S., & Lee, H. (2017). Value prediction network. In *Advances in Neural Information Processing Systems* (pp. 6118–6128).

Ortner, R. (2013). Adaptive aggregation for reinforcement learning in average reward Markov decision processes. *Annals of Operations Research*, *208*(1), 321–336.

Ostrovski, G., Bellemare, M. G., Oord, A., & Munos, R. (2017). Count-based exploration with neural density models. In *International conference on machine learning* (pp. 2721–2730).

Oudeyer, P.-Y., Kaplan, F., & Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, *11*(2), 265–286.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th international conference on machine learning - volume 70* (pp. 2778–2787). JMLR.

Ribas-Fernandes, J., Solway, A., Diuk, C., McGuire, J., Barto, A., Niv, Y., & Botvinick, M. (2011). A Neural Signature of Hierarchical Reinforcement Learning. *Neuron*, *71*(2), 370–379. doi: 10.1016/J.NEURON.2011.05.042

Rosenbaum, D. A., Inhoff, A. W., & Gordon, A. M. (1984). Choosing between movement sequences: A hierarchical editor model. *Journal of Experimental Psychology: General*, *113*(3), 372–393. doi: 10.1037/0096-3445.113.3.372

Rosenbaum, D. A., Kenny, S. B., & Derr, M. A. (1983). *Hierarchical control of rapid movement sequences.* (Vol. 9) (No. 1). US: American Psychological Association. doi: 10.1037/0096-1523.9.1.86

Russell, S. J. (1997). Rationality and intelligence. *Artificial Intelligence*, *94*(1-2), 57–77. doi: 10.1016/S0004-3702(97)00026-X

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, *5*(2), 115 - 135. doi: https://doi.org/10.1016/0004-3702(74)90026-5

Sanborn, S., Bourgin, D. D., Chang, M., & Griffiths, T. L. (2018). Representational efficiency outweighs action efficiency in human program induction. In T. Rogers, M. Rau, X. Zhu, & C. Kalish (Eds.), *Proceedings of the 40th annual meeting of the cognitive science society* (pp. 2400–2405). Austin, TX: Cognitive Science Society.

Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats* (pp. 222–227).

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., … Degris, T. (2017). *The predictron: end-to-end learning and planning.* JMLR.

Şimşek, Ö., & Barto, A. G. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Twenty-first international conference on machine learning - icml '04* (p. 95). New York, New York, USA: ACM Press. doi: 10.1145/1015330.1015353

Solway, A., Diuk, C., Córdova, N., Yee, D., Barto, A. G., Niv, Y., & Botvinick, M. M. (2014). Optimal Behavioral Hierarchy. *PLoS Computational Biology*, *10*(8), e1003779. doi: 10.1371/journal.pcbi.1003779

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1), 181–211.

Taiga, A. A., Courville, A., & Bellemare, M. (2018). Approximate exploration through state abstraction. *ICML Workshop on Exploration in RL.*

Topin, N., Haltmeyer, N., Squire, S., Winder, J., desJardins, M., & MacGlashan, J. (2015). Portable option discovery for automated learning transfer in object-oriented Markov decision processes. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 3856–3864).