UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Markov Decision Processes

Seminar Essay
by Sanjay Chhataru Gupta

Research group Data Science 4 Engineering

Contents

1 Abstract

Reinforcement Learning (RL) is a subfield of machine learning (ML) that focusses on training agents to make sequential decisions in dynamic environments to maximise cumulative rewards. Markov Decision Processes (MDPs) provide a fundamental framework

for modelling and solving RL problems. This report aims to explore the concepts and implications of MDPs in the context of RL, specifically focusing on finite and infinite MDPs.

2 Introduction

In RL, an agent is faced with the problem of learning behaviour through trial-and-error interactions with a dynamic environment [KLM96]. In recent years, RL has received much attention due to its ability to solve complex decision-making problems and its application in many different areas including robotics, healthcare, gaming, and finance [SB98a]. For example, the management of diabetes requires continuous monitoring and adjustment of treatment plans to maintain blood sugar levels within a target range. RL can be used to develop personalised treatment strategies that adapt to the characteristics of the individual patient and optimise long-term health outcomes [YOA+20]. The Markov Decision Process (MDP) is at the core of RL, providing a mathematical framework to describe and solve sequential decision-making [SB98b]. As a powerful framework for modelling real-world scenarios, MDPs capture an environment's Markovian property, where future states depend only on the present state and actions are taken [SB98b]. MDPs enable agents to learn optimal decision-making policies in dynamic environments. By formulating RL problems as MDPs, we can effectively model the interaction between an agent and its environment, allowing the agent to learn and adapts its behaviour to maximise cumulative rewards over time.

Many real-world problems involve uncertainty and incomplete knowledge about the environment. MDPs provide a structured framework to handle uncertainty by incorporating probabilistic transition dynamics, allowing agents to explore different actions and learn from the resulting feedback to make informed decisions [CPC12]. MDPs provide a basis for generalisation and transfer learning in RL [Tay09]. By learning policies in MDPs, agents can transfer their knowledge and experiences to new, similar tasks or environments, accelerating the learning process and enabling more efficient decision-making. MDPs help capture the trade-offs between short-term rewards and long-term goals [SB98b]. By incorporating a discount factor, MDPs allow for balancing immediate rewards with future rewards, enabling agents to make decisions that optimize long-term outcomes. MDPs provide a formal framework for evaluating the performance of RL algorithms and comparing different approaches [Sze11]. The well-defined mathematical properties of MDPs allow for rigorous analysis, convergence guarantees, and theoretical insights into the behaviour and efficiency of RL algorithms.

By studying MDPs, researchers and practitioners gain a deeper understanding of the underlying principles, algorithms, and techniques for solving sequential decision-making problems. This understanding contributes to the development of more effective RL algorithms, better decision-making systems, and advancements in various application domains. The structure of the report will be as follows. Section 3, will give an overview of MDPs which will cover their definition, components, and the distinction between finite and infinite MDPs, highlighting the challenges and extensions associated with

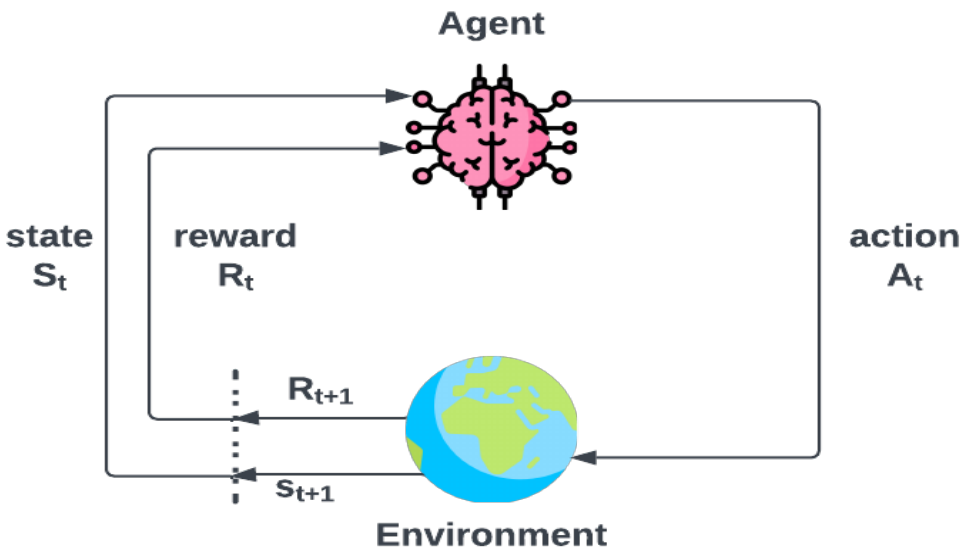

Figure 1: The agent-environment interaction in an MDP.

infinite MDPs. The section 4 will go into solving finite MDPs through policy evaluation and policy improvement. Approximation techniques for infinite MDPs are examined in section 5, with a focus on value function approximation and policy approximation methods. The comparative analysis of finite and infinite MDPs provides insight into the differences in problem formulation and solution methods discussed in section 6.

3 MDP: An Overview

In RL problems, we try to learn how agents make decisions in complex and uncertain environments. As mentioned in section 2, MDPs are meant to be a straightforward problem of learning from interaction to achieve a goal [SB98b]. Figure 1 shows the agent-environment interaction in MDP [SB98b]. The agent is the learner or decision-maker. The environment is the thing with which agents interact, and it includes everything outside of the agent.

A MDP is a tuple of States ($\mathcal{S}$), Actions ($\mathcal{A}$), State transition probabilities ($\mathcal{P}$), Reward function ($\mathcal{R}$), and Discount factor (γ) [SB98b].

- **Initial States** represent different possible configurations or situations of the environment. Each state is a snapshot of the system at a particular time.
- **Actions** are the choices or decisions available to the agent in a given state. These actions can influence the state transition and subsequent rewards.

- **State transition probabilities** define the likelihood of moving from one state to another based on the chosen action. It captures the dynamics of the environment.

$$\mathcal{P}(s'|s,a) = \mathbb{P}\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} \tag{1}$$

- **Reward function** are numerical values that provide feedback to the agent based on its actions and the resulting state transitions. Rewards can be positive, negative, or zero, reflecting the desirability or undesirability of the outcome. Unbounded rewards are theoretically possible but make the learning process more challenging, particularly when working with value iteration or approximation methods, as these algorithms might struggle to converge or encounter numerical stability issues with extremely large or small reward values [ZS21, HLH19].

$$\mathcal{R}(s,a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] \tag{2}$$

- **Discount factor** determines the importance of future rewards compared to immediate rewards. Influence the agent's decision-making by controlling the extent to which future rewards are considered in the present. When the $\gamma \approx 1$, the agent places a higher emphasis on long-term rewards. On the other hand, when the $\gamma \approx 0$, the agent focuses more on obtaining immediate rewards, and the impact of future rewards decreases. In practice, setting $\gamma = 1$ may be useful when the time horizon is infinite because it simplifies the decision-making process by treating all rewards equally.

$$\gamma \in [0, 1]$$

The interaction between agent-environment follows a sequential and iterative process at discrete time steps, t = 0,1,2,3,..... . At each time step, the agent takes action, $A_t \in \mathcal{A}(s)$ based on the current state, $S_t \in \mathcal{S}$, and receives feedback from the environment in the form of rewards, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and new state, S_{t+1}. As a result of the MDP and the agent working together, the following sequence is created:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ...$$

Almost all RL techniques require the estimation of value functions that capture the expected cumulative rewards an agent can achieve under a given policy [SB98b]. For finite-state MDPs with finite action spaces and bounded rewards, the value function is guaranteed to exist [BD23]. This is because the Bellman equations 7, which relate the value of a state to the values of its successor states, converge to a unique solution for any policy. However, for infinite-state MDPs, the existence of a value function can be more complex [WAJ+21]. In cases where the state or action spaces are infinite and unbounded, traditional methods for solving MDPs might not guarantee the existence of a value function. In such situations, techniques like function approximation (e.g., using neural networks) methods are often employed to estimate the value function [Cer23].

Figure 2, demonstrates the modelling of a Paderborn University student as an MDP. For example, consider that students must sit in two classes to pass the course. Students

begin class one, and there is a 40 % chance that they will attend class two, or they will open their phone and begin using Instagram. Once a student has become addicted to Instagram, there is a 80 % chance that the student will self-transition to Instagram. After a few minutes, there is a 20 % probability that students will leave Instagram and return to the rest of the lectures. If a student attends all of the classes, he or she may pass the course and then head to the cafe to relax or have fun. The student will be rewarded for progressing from one state to another based on his or her actions. The sleep state is a representation of the terminal state. The few sequences will be as follows:

- Class 1, Class 2, Pass, Sleep.
- Class 1, Instagram, Class 1, Class 2, Pass, Sleep.
- Class 1, Instagram, Class 1, Class 2, Cafe, Class 2, Pass, Sleep.
- Class 1, Class 2, Sleep.

By modelling the university student as an MDP, we can explore strategies to optimise their decision-making process. The student can learn an optimal policy that balances academic responsibilities, social interactions, and personal well-being, leading to a more fulfilling and successful university experience.

An optimal policy denoted by π_*, is a policy that maximizes the expected cumulative reward over the long run in an MDPs [SB98b]. It specifies the action a to take at each state s to achieve the highest possible expected return. In an MDPs, there can be multiple optimal policies, all of which yield the same maximum expected return. Optimal policies are characterized by their ability to make the best choices given the current state s and the knowledge of the MDP's dynamics and rewards.

3.1 Finite MDPs

A finite MDP has a finite number of elements in the sets of states ($\mathcal{S}$), actions ($\mathcal{A}$), and rewards ($\mathcal{R}$) [SB98b]. Finite MDPs follow the Markov property, which asserts that the future state exclusively depends on the current state and the action taken, regardless of the past history. The Markov property guarantees that the MDP meets the memorylessness property [SWS$^+$20]. The dynamics of the environment, including the state transition probabilities and the reward function, are stagnant. This means that they remain consistent during the learning process and do not change over time.

$$\mathcal{P}(s', r|s, a) = \mathbb{P}\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{3}$$

A policy is a mapping between states to the probabilities of each possible action being chosen. There are two types of value functions: the state value function and the action value function [SB98b]. The state value function denoted $v_\pi(s)$, represents the cumulative expected rewards an agent can obtain starting from a particular state s and following a specific policy π [SB98b]. The function v_π is the state value function of the policy π.

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \tag{4}$$

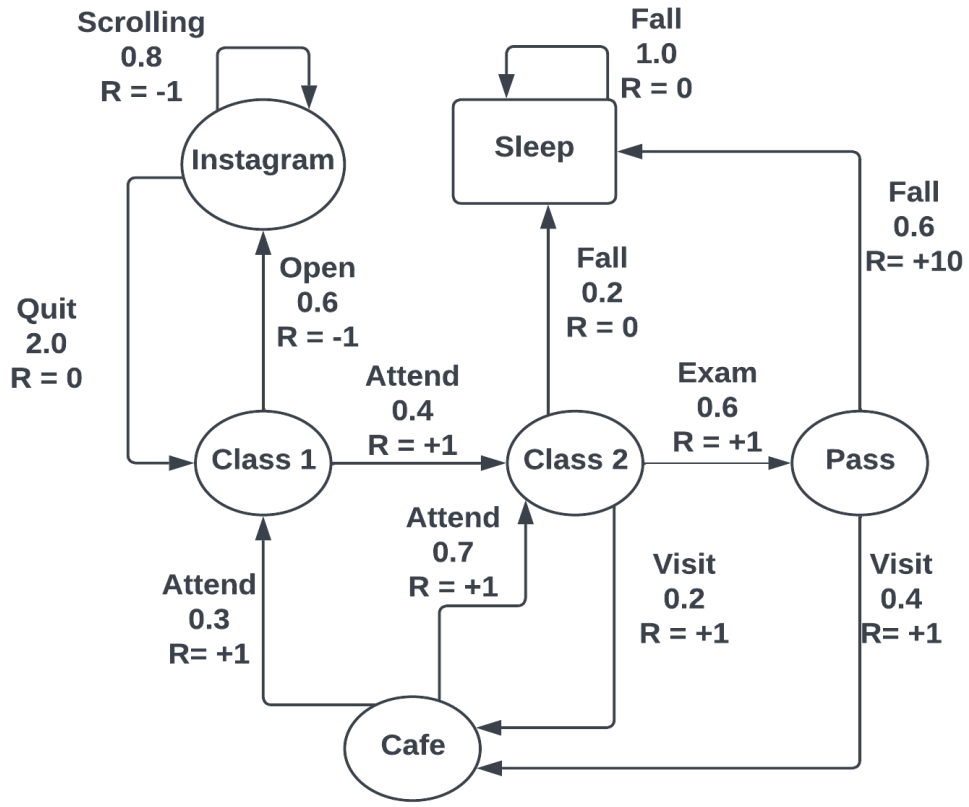

Figure 2: Paderborn University Student as an MDP.

for all $s \in \mathcal{S}$, where $\mathbb{E}_\pi[.]$ represents the expected value of a random variable if the agent follows policy π and t represents any time step.

The action value function is also known as the Q-function and is denoted as $q_\pi(s, a)$. It represents the expected cumulative rewards that an agent can achieve by taking action a in state s and following a specific policy π [SB98b]. In equation 5, q_π the action value function for policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \tag{5}$$

3.2 Infinite MDPs

Infinite MDPs refer to a class of MDPs where the state or action space is infinite or unbounded [GR13]. In Infinite MDPs, the transition probabilities from one state to another are often described using Probability Density Functions (PDFs). PDFs indicate

the probability density of transitioning from a given state to a range of possible successor states, given a specific action. Mathematically, for a state transition from state s to state s' under action a, the transition probability $p(s'|s,a)$ can be represented as a PDFs:

$$p(s', s|a) = f(s', s|a) \tag{6}$$

Where $f(s'|s,a)$ is the PDF that specifies the likelihood of transitioning to state s' when starting from state s and taking action a. Unlike finite MDPs, which have a discrete and finite number of states and actions, infinite MDPs present additional challenges and complexities due to their unbounded nature. In real-world applications, infinite MDPs are frequently encountered when the state or action space is unbounded or infinite.

For example, consider the task of controlling an inverted pendulum to balance it upright. The state of the pendulum can be represented by its angular position and velocity. The action space $\mathcal{A}$ consists of applying torque to the pendulum. The objective is to find a control policy that balances the pendulum in an upright position while minimising energy consumption. In these infinite MDPs, the state space $\mathcal{S}$ is infinite, ranging from $-\pi$ to $+\pi$ for the angular position and $-\infty$ to $+\infty$ for the angular velocity. The action space is also infinite, representing the torque applied to the pendulum. The dynamics of the system are described by ordinary or partial differential equations.

In many real-world scenarios, the state space may be infinite or unbounded, which poses challenges for the approximation and representation of the value function [GR13].

4 Solving Finite MDPs

This section aims to provide a comprehensive understanding of the techniques used to solve finite MDPs. It will cover the foundational concepts of policy evaluation using Bellman equations, iterative methods for value function estimation, and the value iteration algorithm for finding optimal value functions. Additionally, it will explore policy improvement, optimal policies, and the policy iteration algorithm to find the optimal policy.

4.1 Policy Evaluation

Policy Evaluation is the process of determining the value function for a given policy in an MDP [SB98c]. It involves estimating the expected cumulative rewards starting from each state under the given policy.

4.1.1 Bellman equation

Bellman equation provides a recursive relationship between the value function and the immediate rewards and future values [SB98b]. Equation 7 is the Bellman equation for v_π.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right] \tag{7}$$

Where $\pi(a|s)$ represents the probability of taking action a in state s under policy π, and the expectations are subscripted by π to show that they are conditional on π being followed. The Bellman equation for the state value function $v_\pi(s)$ is defined as:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \middle| S_t = s \right] \tag{8}$$

Similarly, the Bellman equation for action value function $q_\pi(s, a)$ is defined as:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \middle| S_t = s, A_t = a \right] \tag{9}$$

Where A_t is the action taken at time step t.

4.1.2 Iterative methods

Iterative methods are commonly used to solve the Bellman equations and estimate the value functions. One such method is iterative policy evaluation, which repeatedly updates the value function estimates until convergence [SB98c]. It starts with an initial estimate of the value function and then iteratively updates the value of each state based on the Bellman equation until the values stabilise. The iterative process of policy evaluation can be summarized as follows [SB98c]:

1. Initialize the value function $V(s)$ arbitrarily for all states s.
2. Iterate until convergence: For each state s, update the value function estimate:

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} \mathcal{P}(s', r|s, a)[r + \gamma V(s')] \tag{10}$$

   for all possible actions a and next states s'.
3. Repeat Step 2 until the value function estimates no longer change significantly.

By repeatedly applying the Bellman equation and updating the value function, iterative policy evaluation gradually refines the value estimates and converges to the true value function for the given policy.

4.1.3 Value iteration algorithm

It combines the ideas of policy evaluation and policy improvement to iteratively update the value function until convergence [SB98c]. It works by repeatedly applying the Bellman optimality equation, which relates the value function of a state to the maximum expected return achievable by following an optimal policy v_*. The algorithm updates the value of each state based on the maximum expected return over all possible actions. The value iteration algorithm can be summarized as follows [SB98c]:

1. Initialize the value function $V(s)$ arbitrarily for all states s.

2. Repeat until convergence: For each state s, update the value function estimate:

$$V(s) \leftarrow \max_a \sum_{s',r} [\mathcal{P}(s', r|s, a)(R(s, a) + \gamma V(s'))] \tag{11}$$

   for all possible actions a and next states s'.

3. Repeat Step 2 until the value function estimates no longer change significantly.

In each iteration, the value of each state s is updated by taking the maximum over the expected returns of all possible actions a. The algorithm continues iterating until the value function estimates converge, meaning that the values no longer change significantly between iterations. Once the value iteration algorithm converges, the resulting value function can be used to derive the optimal policy.

4.2 Policy Improvement

Policy improvement is a step in the process of finding an optimal policy in MDPs. It involves modifying the current policy to improve its expected cumulative reward by making greedy choices based on the estimates of the value function [SB98c]. Policy improvement can be summarized as follows [SB98c]:

1. Given a policy π, evaluate its value function $v_\pi(s)$ for all states s using policy evaluation techniques.

2. For each state s, update the policy π by selecting the action a that maximizes the expected return, based on the value function estimates:

$$\pi'(s) = \arg\max v_\pi(s, a) \tag{12}$$

The policy improvement step ensures that the updated policy π' is at least as good as the current policy π, meaning it either maintains or improves the expected cumulative reward.

5 Approximation techniques for infinite MDPs

The approximation techniques allow us to handle infinite MDPs by reducing computational complexity and enabling efficient learning in infinite or unbounded state and action spaces [SB98d]. By approximating the value functions or policies, we can make RL problems feasible in real-world scenarios where exact tabular representations are infeasible or impractical.

5.1 Value function approximation

Value function approximation is a technique used in RL to estimate the value function of MDPs using a parameterized function [SB98d]. Instead of explicitly storing values for

each state or state-action pair, value function approximation approximates the values using a set of parameters and function approximations. There are several methods for value function approximation:

1. **Linear Function Approximation** represents the value function using a linear combination of features [WJLJ21]. Features can be derived from the state or state-action pairs and are multiplied by corresponding weight parameters. The value function is estimated by adjusting the weights to minimize the prediction error.

2. **Neural Networks** are powerful function approximators commonly used for value function approximation in reinforcement learning. The neural network takes the state or state-action pair as input and produces an estimate of the value function as output [ZPS96]. During training, the neural network's weights are adjusted using gradient-based optimization methods, such as stochastic gradient descent, to minimize the prediction error.

3. **Decision Trees** can be used to approximate the value function by partitioning the state or state-action space into regions based on certain criteria [BD96]. Each region represents a subset of states or state-action pairs, and the value function is approximated with a constant value for each region. Decision trees can be constructed using techniques like the ID3 [Qui86] or the CART [BFOS84] algorithm.

5.2 Policy approximation

Policy approximation is a technique used in RL to approximate the policy function in MDPs using a parameterized function [SB98e]. Instead of explicitly representing the policy as a lookup table or a set of rules, policy approximation uses a function approximator to estimate the policy based on the observed states and actions. There are several methods for policy approximation:

1. **Policy Gradient Methods** are a class of RL algorithms that directly optimize the policy function by using gradient-based optimization [SB98e]. Instead of approximating the value function, policy gradient methods directly search for the optimal policy by iteratively updating the policy parameters. The key idea behind policy gradient methods is to estimate the gradient of the expected cumulative reward with respect to the policy parameters and perform gradient ascent to maximize the reward.

2. **Monte Carlo Tree Search (MCTS)** is a planning algorithm commonly used in policy approximation for decision-making problems with large state and action spaces [SB98f]. MCTS is particularly effective in games and sequential decision-making tasks. MCTS builds a search tree by iteratively expanding and exploring different paths in the search space. At each iteration, it performs four main steps: selection, expansion, simulation, and backpropagation.

Table 1: Comparative analysis of Finite and Infinite MDPs.

Characteristics	**Finite MDPs**	**Infinite MDPs**
The complexity of the environment	Small and discrete state and action space	Infinite or unbounded state or action space
Convergence	Optimal policy	Pose challenges
Resource Requirements	Computationally efficient	Computational complexity
Trade-offs in Modeling	Concise and interpretable	Complex
Tasks	Grid-world navigation, Puzzle-solving, Board games	Autonomous driving, Robotic control, Economic decision-making

Both policy gradient methods and MCTS are powerful techniques in policy approximation. Policy gradient methods directly optimize the policy based on gradient ascent, while MCTS focuses on efficient exploration and exploitation by building a search tree.

6 Comparative Analysis

Finite and infinite MDPs have different characteristics and are suited for different types of applications. Table 1 represents the comparative analysis of the applications of finite and infinite MDPs.

Finite MDPs are well-suited for problems with a small and discrete state and action space, while infinite MDPs are suitable for problems with an Infinite or unbounded state or action space. Finite MDPs allow for exact solutions and convergence to the optimal policy. Infinite MDPs pose challenges in convergence due to their infinite state or action spaces. Finite MDPs can be solved using exact methods, making them computationally efficient for small-scale problems. Infinite MDPs with unbounded state spaces require approximation methods due to the computational complexity involved. Finite MDPs provide a more concise and interpretable model, making it easier to reason about the system's behaviour and policy choices. Infinite MDPs require more sophisticated modelling techniques to handle unbounded states or actions, making the models more complex.

7 Conclusion

In conclusion, the study of MDPs, finite and infinite MDPs in RL provides a robust framework for modelling and solving complex sequential decision-making problems. Through an understanding of MDP formulation, properties, and solution methods, re-

searchers and practitioners can effectively address a wide range of real-world challenges. The comparative analysis between finite and infinite MDPs highlights the trade-offs, considerations, and diverse application domains associated with each type. Ongoing research in approximation techniques and exploration-exploitation balancing continues to enhance the capabilities of MDPs, enabling them to tackle increasingly complex RL problems. Overall, MDPs serve as a versatile foundation for optimal decision-making, holding great promise for future advancements in the field.

References

[BD96] Craig Boutilier and Richard Dearden. Approximate value trees in structured dynamic programming. In Lorenza Saitta, editor, *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 54–62. Morgan Kaufmann, 1996.

[BD23] El Akraoui Bouchra and Cherki Daoui. Solving finite-horizon discounted non-stationary mdps. *Folia Oeconomica Stetinensia*, 23(1):1–15, 3923.

[BFOS84] L. Breiman, J. H. Freidman, Richard A. Olshen, and C. J. Stone. Cart: Classification and regression trees. In *Classification and Regression Trees*, 1984.

[Cer23] Cristiano Cervellera. Optimized ensemble value function approximation for dynamic programming. *European Journal of Operational Research*, 309(2):719–730, 2023.

[CPC12] Vincenzo Cannella, Roberto Pirrone, and Antonio Chella. Comprehensive uncertainty management in mdps. In Antonio Chella, Roberto Pirrone, Rosario Sorbello, and Kamilla R. Johannsdottir, editors, *Biologically Inspired Cognitive Architectures 2012 - Proceedings of the Third Annual Meeting of the BICA Society, Palermo, Sicily, Italy, October 31 - November 3, 2012*, volume 196 of *Advances in Intelligent Systems and Computing*, pages 89–94. Springer, 2012.

[DMM+22] Christoph Dann, Yishay Mansour, Mehryar Mohri, Ayush Sekhari, and Karthik Sridharan. Guarantees for epsilon-greedy reinforcement learning with function approximation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 4666–4689. PMLR, 2022.

[GKS21] Ketika Garg, Christopher T. Kello, and Paul E. Smaldino. Individual exploration and selective social learning: Balancing exploration-exploitation trade-offs in collective foraging. *bioRxiv*, 2021.

[GR13] Frédérick Garcia and Emmanuel Rachelson. *Markov Decision Processes*, chapter 1, pages 1–38. John Wiley and Sons, Ltd, 2013.

[HLH19] Rakhoon Hwang, Hanjin Lee, and Hyung Ju Hwang. Option compatible reward inverse reinforcement learning. *CoRR*, abs/1911.02723, 2019.

[IFSR+23] Sardor Israilov, Li Fu, Jesús Sánchez-Rodríguez, Franco Fusco, Guillaume Allibert, Christophe Raufaste, and Médéric Argentina. Reinforcement learning approach to control an inverted pendulum: A general framework for educational purposes. *PLOS ONE*, 18(2):1–15, 02 2023.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4:237–285, 1996.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[SB98a] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*, chapter 1, pages 1–22. Adaptive computation and machine learning. MIT Press, 1998.

[SB98b] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*, chapter 3, pages 47–71. Adaptive computation and machine learning. MIT Press, 1998.

[SB98c] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*, chapter 4, pages 73–90. Adaptive computation and machine learning. MIT Press, 1998.

[SB98d] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*, chapter 9, pages 197–242. Adaptive computation and machine learning. MIT Press, 1998.

[SB98e] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*, chapter 13, pages 321–338. Adaptive computation and machine learning. MIT Press, 1998.

[SB98f] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*, chapter 8, pages 159–193. Adaptive computation and machine learning. MIT Press, 1998.

[SWS+20] Chengchun Shi, Runzhe Wan, Rui Song, Wenbin Lu, and Ling Leng. Does the markov decision process fit the data: Testing for the markov property in sequential decision making. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8807–8817. PMLR, 2020.

[Sze11] Csaba Szepesvári. *Reinforcement Learning Algorithms for MDPs*, chapter 1, pages 1–73. John Wiley and Sons, Ltd, 2011.

[Tay09] Matthew E. Taylor. *Transfer in Reinforcement Learning Domains*, volume 216 of *Studies in Computational Intelligence*. Springer, 2009.

[WAJ+21] Gellért Weisz, Philip Amortila, Barnabás Janzer, Yasin Abbasi-Yadkori, Nan Jiang, and Csaba Szepesvári. On query-efficient planning in mdps under linear realizability of the optimal state-value function. *CoRR*, abs/2102.02049, 2021.

[WJLJ21] Chen-Yu Wei, Mehdi Jafarnia-Jahromi, Haipeng Luo, and Rahul Jain. Learning infinite-horizon average-reward mdps with linear function approximation. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 3007–3015. PMLR, 2021.

[YOA+20] Taku Yamagata, Aisling Ann O'Kane, Amid Ayobi, Dmitri S. Katz, Katarzyna Stawarz, Paul Marshall, Peter A. Flach, and Raúl Santos-Rodríguez. Model-based reinforcement learning for type 1 diabetes blood glucose control. In Ester Zumpano, Andrea Tagarelli, Carmela Comito, Sergio Greco, Pierangelo Veltri, Agusti Solanas, Inmaculada Mora, and Miquel Sànchez-Marrè, editors, *Proceedings of the First International AAI4H - Advances in Artificial Intelligence for Healthcare Workshop co-located with the 24th European Conference on Artificial Intelligence (ECAI 2020), Santiago de Compostela, Spain, September 4, 2020*, volume 2820 of *CEUR Workshop Proceedings*, pages 72–77. CEUR-WS.org, 2020.

[ZCPZ19] Yu Zhang, Peixiang Cai, Changyong Pan, and Subing Zhang. Multi-agent deep reinforcement learning-based cooperative spectrum sensing with upper confidence bound exploration. *IEEE Access*, 7:118898–118906, 2019.

[ZPS96] R. Zoppoli, T. Parisini, and M. Sanguineti. Neural approximators for functional optimization. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 3, pages 3290–3293 vol.3, 1996.

[ZS21] Vincent Zhuang and Yanan Sui. No-regret reinforcement learning with heavy-tailed rewards. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 3385–3393. PMLR, 2021.