Week #14: Decision-Making and Collaboration

Notes by: Francisco Richter

May 15, 2025

Overview

Multi-agent systems—where multiple autonomous agents interact in a shared environment—are central to artificial intelligence and operations research. When randomness influences system dynamics, *stochastic processes* provide the appropriate mathematical framework. This lecture offers a rigorous exposition of stochastic processes in multi-agent decision-making. We first review key models, including Markov Decision Processes (MDPs), stochastic games (Markov games), and Decentralized Partially Observable MDPs (Dec-POMDPs). Next, we discuss solution methodologies, reinforcement learning approaches, and advanced topics such as exploration-exploitation tradeoffs and bias. Finally, we introduce a comprehensive multi-agent framework that defines agents, tasks, tools, crews, processes, and collaboration—equipping you with both theoretical foundations and practical design principles.

---

1 Fundamentals of Sequential Decision-Making

MDPs provide a mathematical model for a single agent interacting with a stochastic environment. In an MDP the agent chooses actions in a series of states so as to maximize the expected total discounted reward.

Definition. Markov Decision Process (MDP)

An MDP is a tuple

$$\mathcal{M} = \langle S, A, P, R, \gamma \rangle,$$

where:

- S is a finite set of states.
- A is a finite set of actions.
- $P(s' \mid s, a)$ is the probability of transitioning from state s to state s' when action a is taken, satisfying

$$\sum_{s' \in S} P(s' \mid s, a) = 1.$$

- $R(s, a)$ is the immediate reward received when action a is executed in state s.
- $\gamma \in [0, 1)$ is the discount factor, which reflects the fact that immediate rewards are typically more valuable than future rewards.

A *policy* π is a mapping $\pi : S \to A$ (or more generally $\pi : S \to \Delta(A)$ for randomized strategies). The goal is to find an optimal policy π^* that maximizes the expected total discounted reward.

Definition. Value Functions

For a policy π, the *state-value function* is defined as

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s,\ a_t = \pi(s_t)\right].$$

Similarly, the *action-value function* is

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a)\, V^{\pi}(s').$$

The optimal state-value function, defined as

$$V^*(s) = \max_{\pi} V^{\pi}(s),$$

satisfies the *Bellman optimality equation*:

$$V^*(s) = \max_{a \in A}\Big\{R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a)\, V^*(s')\Big\}.$$

Theorem 1.1 (Existence and Uniqueness of V^*)**.** *For any finite MDP with $\gamma \in [0,1)$, the Bellman operator*

$$(TV)(s) = \max_{a \in A}\Big\{R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a)\, V(s')\Big\}$$

is a contraction mapping. Hence, it has a unique fixed point V^* and iterative application (value iteration) converges to V^*.

Proof Sketch. Since $\gamma < 1$, for any two value functions V and W, we have

$$\|TV - TW\|_{\infty} \le \gamma \|V - W\|_{\infty}.$$

Thus, T is a contraction with contraction factor γ in the supremum norm. By Banach's fixed-point theorem, there exists a unique fixed point V^* such that $TV^* = V^*$, and the sequence

$$V_{k+1} = TV_k$$

converges to V^* for any initial function V_0. □

This theorem guarantees that if you repeatedly apply the Bellman update to any initial guess of the value function (even an arbitrary one), you will converge to the unique optimal value function V^*. In practical terms, it means that the method known as *value iteration* is reliable and will eventually yield the optimal values for all states. The contraction property (with factor γ) ensures that errors decrease exponentially with each iteration. This theoretical result underpins many dynamic programming algorithms in reinforcement learning.

Example 1.1 (Gridworld)**.** *Consider an agent navigating a grid where each cell represents a state. The agent may move in the four cardinal directions (actions). Suppose each move incurs a cost (e.g., -1 reward) except reaching a goal state where a positive reward is obtained. In this MDP, the value iteration algorithm will update the value of each cell based on the expected discounted rewards of moving toward the goal. The convergence guaranteed by the theorem tells us that, after enough iterations, the computed values will accurately reflect the best achievable rewards from every cell.*

2 Extending to Multiple Agents: Stochastic Games

When several agents interact, each with potentially different objectives, we use the framework of *stochastic games* (or Markov games).

Definition. Stochastic Game

An n-player stochastic game is a tuple

$$\mathcal{G} = \langle n, S, \{A_i\}_{i=1}^n, P, \{R_i\}_{i=1}^n, \gamma \rangle,$$

where

- n is the number of agents.
- S is the finite set of states.
- For each agent i, A_i is its finite action set; the joint action is $a = (a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$.
- $P(s' \mid s, a)$ is the state transition probability function.
- For each agent i, $R_i(s, a)$ is the reward received.
- $\gamma \in [0, 1)$ is the common discount factor.

Each agent selects a policy $\pi_i : S \to A_i$ (or randomized version). The value for agent i under a joint policy $\pi = (\pi_1, \ldots, \pi_n)$ is

$$V_i^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_t) \,\middle|\, s_0 = s,\ a_t = (\pi_1(s_t), \ldots, \pi_n(s_t))\right].$$

Definition. Nash Equilibrium in Stochastic Games

A joint policy $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$ is a Nash equilibrium if, for each agent i and for every alternative policy $\hat{\pi}_i$,

$$V_i^{(\pi_i^*, \pi_{-i}^*)}(s) \geq V_i^{(\hat{\pi}_i, \pi_{-i}^*)}(s) \quad \text{for all } s \in S,$$

where π_{-i}^* denotes the policies of all agents except i.

Example 2.1 (Zero-Sum Game on a Grid)**.** *Imagine a two-player grid game where one agent's gain is exactly the other's loss. The state includes both players' positions, and each chooses moves simultaneously. In a zero-sum setting, if player 1's reward is $R(s, a_1, a_2)$, then player 2's reward is $-R(s, a_1, a_2)$. The solution concept is a minimax equilibrium.*

In zero-sum games the Bellman equation is replaced by a minimax formulation:

$$V^*(s) = \max_{a_1 \in A_1} \min_{a_2 \in A_2} \left\{ R(s, a_1, a_2) + \gamma \sum_{s'} P(s' \mid s, a_1, a_2)\, V^*(s') \right\}.$$

3 Decentralized Partially Observable MDPs (Dec-POMDPs)

In many real-world multi-agent systems, agents do not have full state information. The Dec-POMDP framework models cooperative agents acting under partial observability.

Definition. Dec-POMDP

A Decentralized POMDP is defined as a tuple

$$\mathcal{D} = \langle n, S, \{A_i\}_{i=1}^n, P, \{O_i\}_{i=1}^n, O, R, \gamma \rangle,$$

where

- n is the number of agents.
- S is the finite set of states.
- For each agent i, A_i is the action set.
- $P(s' \mid s, a_1, \ldots, a_n)$ is the state transition function.
- For each agent i, O_i is its set of observations.
- $O(o_1, \ldots, o_n \mid s', a_1, \ldots, a_n)$ is the joint observation probability function.
- $R(s, a_1, \ldots, a_n)$ is the common reward function (assuming a cooperative task).
- $\gamma \in [0, 1)$ is the discount factor.

Each agent's policy now depends on its own observation history. Although Dec-POMDPs are a very general and expressive framework, they are known to be **NEXP**-complete to solve optimally even for a small number of agents.

Example 3.1 (Two-Agent Tiger Problem)**.** *Extend the classical Tiger Problem to two agents. Each agent observes a noisy signal about which door conceals the tiger. They must coordinate (without communication during execution) to decide who opens which door so as to avoid the tiger. The agents' decentralized observations force them to maintain separate histories and design joint policies in advance.*

4 A Framework for Multi-Agent Collaboration

In many applications we need to structure interactions between agents. We formalize a framework that decomposes a multi-agent system into agents, tasks, tools, and processes.

Agents and Tasks

We represent each agent i by a tuple

$$A_i = (R_i,\, G_i,\, T_i,\, M_i,\, B_i,\, D_i),$$

where:

- R_i is the agent's role,
- G_i is its goal,
- T_i is the set of tools available,
- M_i represents its memory or knowledge,
- B_i is a backstory or prior experience,
- D_i indicates its capability for delegation.

A task is defined by

$$\tau = (D_\tau,\, E_\tau,\, T_\tau,\, A_\tau),$$

where:

- D_τ is a precise description,
- E_τ is the expected output,
- T_τ are the required tools,
- A_τ is the agent (or crew) assigned.

Processes and Collaboration

A *process* is a sequence of tasks,

$$P = (\tau_1,\, \tau_2,\, \ldots,\, \tau_m),$$

with overall process time

$$T_P = \sum_{i=1}^{m} T_c(\tau_i),$$

where $T_c(\tau_i)$ is the completion time of task τ_i.

Collaboration is modeled by a function

$$\mathcal{C} : C \times T \times M \to C,$$

where a crew C is a set of agents and $\mathcal{C}$ updates their states based on task requirements and shared memory. Metrics such as crew efficiency

$$\eta(C) = \frac{1}{|C|} \sum_{i \in C} P(G_i \mid T_i, M_i)$$

capture how well the agents' individual capabilities combine to achieve a common goal.

5 Advanced Topics and Concluding Remarks

Randomized Strategies and Equilibrium Selection

In multi-agent settings, equilibrium strategies are often mixed (randomized) to avoid predictability. For example, in Rock–Paper–Scissors the only Nash equilibrium is for each player to randomize uniformly over the three actions. In learning algorithms, exploration is achieved by annealing an exploration parameter (e.g., ε in ε-greedy methods).

Complexity and Learning Challenges

While MDPs admit efficient dynamic programming solutions, general stochastic games and Dec-POMDPs are computationally intractable (PPAD-complete or NEXP-complete). This motivates the use of approximate methods such as multi-agent reinforcement learning (MARL). Approaches include:

- **Independent Learning:** Each agent treats others as part of the environment.
- **Centralized Training with Decentralized Execution:** Agents learn a joint value function (or critic) but deploy individual policies.
- **Equilibrium-Based Methods:** Algorithms such as Nash Q-learning incorporate game-theoretic equilibrium computations into the update rules.

Final Remarks

Stochastic processes in multi-agent systems form a rich and challenging area that unifies concepts from MDPs, game theory, and distributed decision-making. This lecture has presented a formal yet intuitive framework—starting from single-agent models and progressing to the decentralized and multi-agent setting—equipping you with both theoretical foundations and practical design principles.

References

1. Sutton, R. S. & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
2. Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.
3. Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2), 286–295.
4. Shapley, L. S. (1953). Stochastic Games. *Proceedings of the National Academy of Sciences*, 39(10), 1095–1100.
5. Littman, M. L. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the 11th International Conference on Machine Learning* (pp. 157–163).
6. Hu, J. & Wellman, M. P. (2003). Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4, 1039–1069.
7. Oliehoek, F. A. & Amato, C. (2016). *A Concise Introduction to Decentralized POMDPs*. Springer.
8. Bernstein, D. S., Givan, R., Immerman, N. & Zilberstein, S. (2002). The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4), 819–840.