

L09 Stochastic Games (Markov Decision Processes).

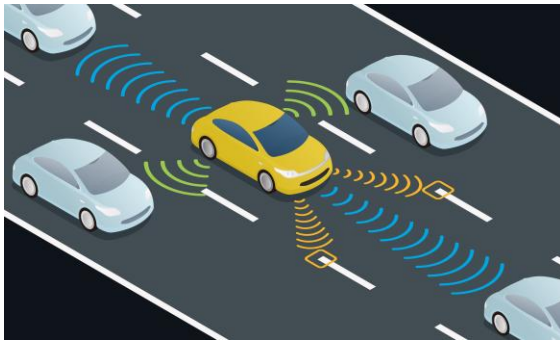
CS 280 Algorithmic Game Theory
Ioannis Panageas

Multi-agent systems and RL

Decentralized systems

Individual interests: cooperation and competition

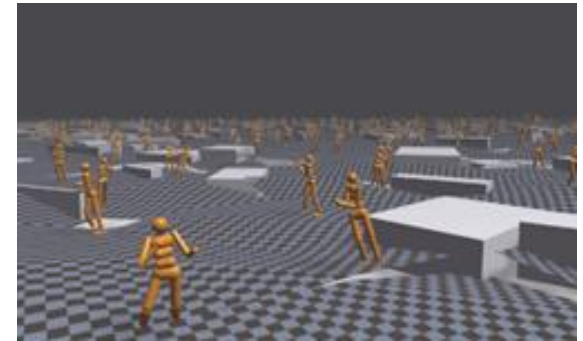
Distributed optimization



Self-driving cars



Auctions



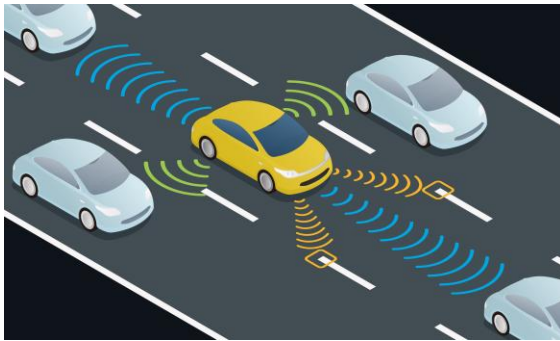
Robotics

Multi-agent systems and RL

Decentralized systems

Individual interests: cooperation and competition

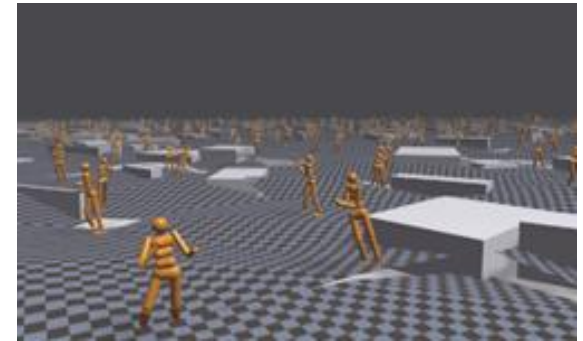
Distributed optimization



Self-driving cars



Auctions



Robotics

How these systems evolve? Predictions?

Markov Games

Markov games model interaction among agents over time.



Player 1



Player 2



Player n

n **players** interact with an environment.

Markov Games

Markov games model interaction among agents over time.



Player 1



Player 2

⋮



Player n

The environment is in state $s \in \mathcal{S}$.

Markov Games

Markov games model interaction among agents over time.



Player 1



Player 2

⋮



Player n

Player i chooses $a_i \in \mathcal{A}_i$.

Actions are chosen simultaneously.

Markov Games

Markov games model interaction among agents over time.



Player 1

Player i receives $r_i(s, a) \in [-1, 1]$.



Player 2

$$s' \sim \mathbb{P}(\cdot \mid s, a)$$

⋮



Player n

The environment moves to the next state.

Markov Games

Markov games model interaction among agents over time.



Player 1



Player 2

⋮



Player n

For a policy profile π and H rounds,

$$V_{i,H}^{\pi}(s) = \mathbb{E}_{\pi, \mathbb{P}} \left[\sum_{t=0}^{H-1} r_i(s_t, a^t) \mid s_0 = s \right].$$

Markov Games

Markov games model interaction among agents over time.



Player 1



Player 2

⋮



Player n

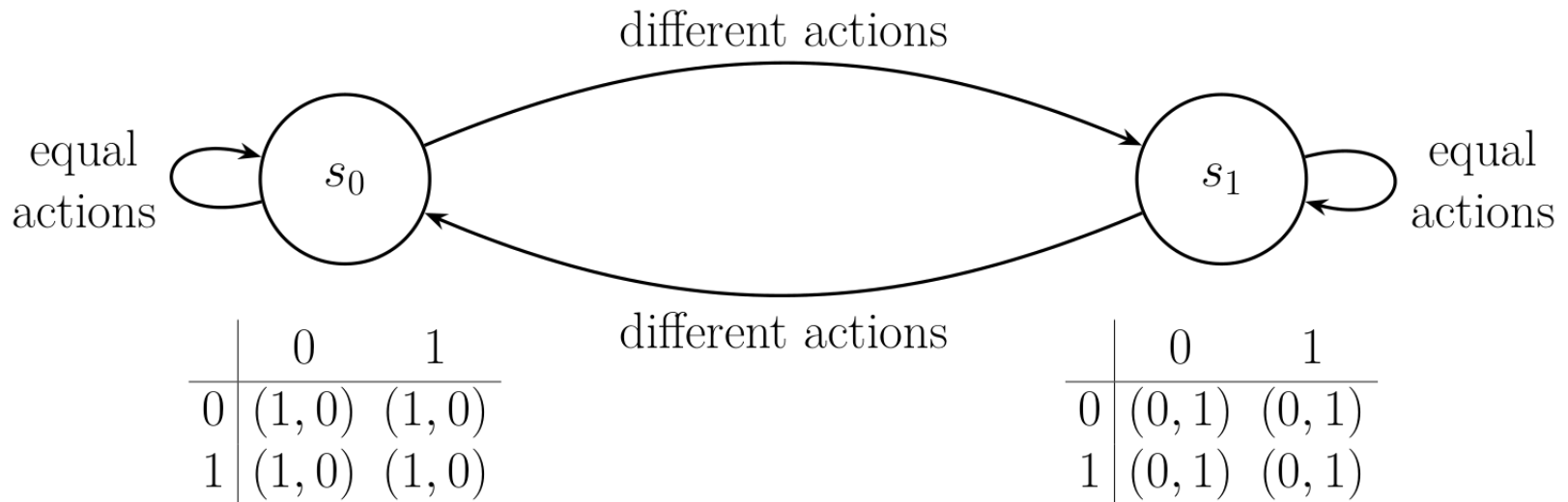
With discount $0 \leq \gamma < 1$,

$$V_i^\pi(s) = \mathbb{E}_{\pi, \mathbb{P}} \left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a^t) \mid s_0 = s \right].$$

Future rewards receive weight γ^t .

An example

Each player chooses 0 or 1. Equal actions keep the current state; different actions switch states.



At s_0 the rewards are (1, 0); at s_1 they are (0, 1).

n -player Markov game: Formal definition

- $\mathcal{N} = \{1, \dots, n\}$: a finite set of players.

n -player Markov game: Formal definition

- $\mathcal{N} = \{1, \dots, n\}$: a finite set of players.
- $\mathcal{S}$: a finite set of states.

n -player Markov game: Formal definition

- $\mathcal{N} = \{1, \dots, n\}$: a finite set of players.
- $\mathcal{S}$: a finite set of states.
- $\mathcal{A}_k$: finite actions for player k . Joint actions: $\mathcal{A} = \prod_{k=1}^n \mathcal{A}_k$.

n -player Markov game: Formal definition

- $\mathcal{N} = \{1, \dots, n\}$: a finite set of players.
- $\mathcal{S}$: a finite set of states.
- $\mathcal{A}_k$: finite actions for player k . Joint actions: $\mathcal{A} = \prod_{k=1}^n \mathcal{A}_k$.
- $r_k : \mathcal{S} \times \mathcal{A} \rightarrow [-1, 1]$: player k 's reward.

n -player Markov game: Formal definition

- $\mathcal{N} = \{1, \dots, n\}$: a finite set of players.
- $\mathcal{S}$: a finite set of states.
- $\mathcal{A}_k$: finite actions for player k . Joint actions: $\mathcal{A} = \prod_{k=1}^n \mathcal{A}_k$.
- $r_k : \mathcal{S} \times \mathcal{A} \rightarrow [-1, 1]$: player k 's reward.
- $\mathbb{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$: next-state distribution.

n -player Markov game: Formal definition

- $\mathcal{N} = \{1, \dots, n\}$: a finite set of players.
- $\mathcal{S}$: a finite set of states.
- $\mathcal{A}_k$: finite actions for player k . Joint actions: $\mathcal{A} = \prod_{k=1}^n \mathcal{A}_k$.
- $r_k : \mathcal{S} \times \mathcal{A} \rightarrow [-1, 1]$: player k 's reward.
- $\mathbb{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$: next-state distribution.
- $\gamma \in [0, 1)$: the discount factor.
- $\boldsymbol{\rho} \in \Delta(\mathcal{S})$: the initial-state distribution.

- *Single agent RL*

The framework

A finite discounted Markov decision process has:

- finite states $\mathcal{S}$ and actions $\mathcal{A}$;
- rewards $r : \mathcal{S} \times \mathcal{A} \rightarrow [-1, 1]$;
- transitions $P(\cdot \mid s, a) \in \Delta(\mathcal{S})$;
- discount factor $0 \leq \gamma < 1$;
- initial distribution $\rho \in \Delta(\mathcal{S})$.

For every (s, a) , $P(s' \mid s, a) \geq 0$ and $\sum_{s' \in \mathcal{S}} P(s' \mid s, a) = 1$.

Definitions

A stationary Markov policy is a map $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$.

$$V^\pi(s) = \mathbb{E}_{\pi, P} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right].$$

$$V^\pi(\rho) = \sum_s \rho(s) V^\pi(s), \quad \text{goal: } \max_{\pi} V^\pi(\rho).$$

Definitions

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

- The maximum may range over all randomized, history-dependent policies.
- A **deterministic stationary** policy is optimal from every state.
- Over randomized stationary policies, $V^{\pi}(\rho)$ **need not be concave** in π .

Example

The state is the **current grid cell**. Actions move north, south, east or west. The dark cell is blocked.

Rewards are zero away from the **goal**. At the **goal**, the agent receives 1 and the episode ends.

Optimal values

0.729	0.81	0.9	★
0.656		0.81	0.9
0.590	0.656	0.729	0.81

An optimal policy

→	→	→	★
↑		↑	↑
↑	→	↑	↑

Which discount factor gives these values?

Bellman operator

For a value vector $W \in \mathbb{R}^{\mathcal{S}}$, the Bellman operator is

$$(\mathcal{T}W)(s) = \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) W(s') \right\}.$$

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

V^* is the unique fixed point of $\mathcal{T}$.

Bellman operator

Conditioning on the first action and next state gives

$$V^*(s) = \max_a \left\{ r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^*(s') \right\}.$$

We will prove $\|\mathcal{T}V - \mathcal{T}W\|_\infty \leq \gamma \|V - W\|_\infty$.

Since $\gamma < 1$, the **Banach fixed-point theorem** gives uniqueness. Here $\|z\|_\infty = \max_s |z(s)|$.

Bellman operator

Fix a state s and write

$$Q_V(s, a) = r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V(s').$$

$$|(\mathcal{T}V)(s) - (\mathcal{T}W)(s)| = \left| \max_a Q_V(s, a) - \max_a Q_W(s, a) \right|.$$

Bellman operator

For finite vectors x, y , $\left| \max_a x_a - \max_a y_a \right| \leq \max_a |x_a - y_a|$.

$$\begin{aligned} |(\mathcal{T}V)(s) - (\mathcal{T}W)(s)| &\leq \max_a |Q_V(s, a) - Q_W(s, a)| \\ &= \gamma \max_a \left| \sum_{s'} P(s' \mid s, a) (V(s') - W(s')) \right|. \end{aligned}$$

Bellman operator

Let a^* maximize x_a . Then

$$\max_a x_a - \max_a y_a \leq x_{a^*} - y_{a^*} \leq \max_a |x_a - y_a|.$$

Exchanging x and y gives the reverse bound, so

$$\left| \max_a x_a - \max_a y_a \right| \leq \max_a |x_a - y_a|.$$

Bellman operator

Transition probabilities are nonnegative and sum to one.

$$\begin{aligned} \left| \sum_{s'} P(s' \mid s, a)(V(s') - W(s')) \right| &\leq \sum_{s'} P(s' \mid s, a) |V(s') - W(s')| \\ &\leq \|V - W\|_{\infty}. \end{aligned}$$

For every state s , $|(\mathcal{T}V)(s) - (\mathcal{T}W)(s)| \leq \gamma \|V - W\|_{\infty}$.

Bellman operator

Taking the maximum over s gives

$$\|\mathcal{T}V - \mathcal{T}W\|_{\infty} \leq \gamma \|V - W\|_{\infty}.$$

- Repeated applications converge to V^* .
- The iteration count depends on $1/(1 - \gamma)$, which may be exponentially large in its binary encoding.
- For a rational finite MDP, linear programming computes an optimal policy in polynomial time.

Value Iteration

Start with any value vector V_0 . At iteration k , apply the Bellman operator:

$$V_{k+1} = \mathcal{T}V_k.$$

$$\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty.$$

Value Iteration

A greedy policy for V_k chooses

$$\pi_k(s) \in \arg \max_a \left\{ r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V_k(s') \right\}.$$

With $V_0 = 0$ and $|r| \leq 1$,

$$\|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1 - \gamma}.$$

For $0 < \gamma < 1$, value error ε follows once

$$k \geq \max \left\{ 0, \left\lceil \frac{\log(1/[\varepsilon(1 - \gamma)])}{\log(1/\gamma)} \right\rceil \right\}.$$

Policy Iteration

Start with a deterministic stationary policy π_0 . Repeat:

$$\text{Evaluate: } (I - \gamma P_{\pi_k})V^{\pi_k} = r_{\pi_k}.$$

$$\text{Improve: } \pi_{k+1}(s) \in \arg \max_a Q_{V^{\pi_k}}(s, a).$$

P_{π_k} and r_{π_k} are the transition matrix and reward vector under π_k .

Keep the current action whenever it is a maximizer. Stop when $\pi_{k+1} = \pi_k$.

- *Markov games: Solution concepts*

Solution Concept: Nash equilibrium

Each player k chooses a **policy** that specifies a distribution over actions.

- **Stationary Markov**: depends only on the current state.
- **Time-dependent Markov**: depends on the current state and time.
- **History-dependent**: may depend on the observed history.

Solution Concept: Nash equilibrium

We focus on stationary randomized policies:

$$\pi_k : \mathcal{S} \rightarrow \Delta(\mathcal{A}_k), \quad a_k^t \sim \pi_k(\cdot \mid s_t).$$

The policy profile is $\pi = (\pi_1, \dots, \pi_n)$.

Each player k aims to maximize $V_k^\pi(\boldsymbol{\rho})$.

Solution Concept: Nash equilibrium

An ε -Nash equilibrium is a policy profile π^* such that no player can gain more than ε by changing only their own policy.

$$V_k^{\pi^*}(\boldsymbol{\rho}) \geq V_k^{(\pi'_k, \pi_{-k}^*)}(\boldsymbol{\rho}) - \varepsilon \quad \text{for every } k, \pi'_k.$$

Solution Concept: Nash equilibrium

An ε -Nash equilibrium is a policy profile π^* such that no player can gain more than ε by changing only their own policy.

$$V_k^{\pi^*}(\boldsymbol{\rho}) \geq V_k^{(\pi'_k, \pi_{-k}^*)}(\boldsymbol{\rho}) - \varepsilon \quad \text{for every } k, \pi'_k.$$

Under a stationary profile, players draw their actions independently given the current state:

$$\Pr(a \mid s) = \prod_{k=1}^n \pi_k(a_k \mid s).$$

Solution Concept: Nash equilibrium

Fix the stationary policies π_{-k} of all players except k .

Player k then faces a single-agent MDP.

- Average the reward and transition law over the opponents' actions.
- A best response can be stationary and deterministic.
- At equilibrium, every player is an ε -best response.

Solution Concept: Nash equilibrium

Theorem (Fink, 1964).

Every finite discounted stochastic game has a Nash equilibrium in stationary randomized policies.

The same equilibrium policies work from every initial state.

Players may need to randomize.

This generalizes mixed Nash equilibrium in normal-form games.

The bad news

A normal-form game is a Markov game with **one state** that always transitions to itself.

$$V_k^\pi = \frac{u_k(\pi)}{1 - \gamma}.$$

The stationary Nash equilibria are **exactly** the mixed Nash equilibria of the original game.

The bad news

Markov games inherit the **difficulty** of computing Nash equilibria in normal-form games.

Finding an ε -Nash equilibrium in stationary policies is **PPAD-hard** for inverse-polynomial ε , even with two players and one state.

The one-state reduction already gives this **hardness**.
Chen, Deng and Teng (2006).

The bad news

General-sum Markov games are computationally difficult.

We next consider two-player zero-sum games.

- The players' rewards sum to zero.
- A minimax value exists.
- A contraction gives a method for approximating that value.

- *Two-player zero sum
Markov games*

2-player zero-sum Markov games

- Player 1 chooses $a \in \mathcal{A}$ and minimizes.
- Player 2 chooses $b \in \mathcal{B}$ and maximizes.
- Let $r = r_2 = -r_1$ denote the maximizer's reward.

$$V^{\pi_1, \pi_2}(\rho) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, b_t) \mid s_0 \sim \rho \right].$$

2-player zero-sum Markov games

At each state, the players choose actions simultaneously. The current reward is given by a matrix.

$$r(s_0, \cdot, \cdot) = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}, \quad r(s_1, \cdot, \cdot) = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

The transition probabilities $P(s' \mid s, a, b)$ specify the next state. Here the available action sets may depend on s .

2-player zero-sum Markov games

For every finite, discounted, two-player zero-sum stochastic game,

$$\min_{\pi_1} \max_{\pi_2} V^{\pi_1, \pi_2}(\rho) = \max_{\pi_2} \min_{\pi_1} V^{\pi_1, \pi_2}(\rho).$$

Both players have optimal stationary randomized policies. Their value is denoted by $V^*(\rho)$.

Shapley (1953).

2-player zero-sum Markov games

- The **value** is unique, but optimal policies need not be.
- Reversing the order of optimization over policies leaves the value unchanged.
- The payoff function **need not be convex-concave** in policy probabilities.
- The proof uses a **contraction** of value vectors.

Polynomial-time value approximation in the full binary input size **remains open** in general. **Quasipolynomial-time** value approximation is now known. Soleymani–Farina (2026).

2-player zero-sum Markov games

For a payoff matrix M , let

$$\text{val}(M) = \min_{x \in \Delta(\mathcal{A})} \max_{y \in \Delta(\mathcal{B})} x^\top M y.$$

$$\text{val} \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} = 0.$$

Shapley's proof replaces a maximum over actions with the **value of a matrix game**.

2-player zero-sum Markov games

For matrices of the same size,

$$|\text{val}(M) - \text{val}(N)| \leq \max_{a,b} |M_{ab} - N_{ab}|.$$

For a continuation-value vector V , define

$$M_s(V)_{ab} = r(s, a, b) + \gamma \sum_{s'} P(s' \mid s, a, b) V(s'), \quad (\mathcal{T}V)(s) = \text{val}(M_s(V)).$$

2-player zero-sum Markov games

$$\begin{aligned} |(\mathcal{T}V)(s) - (\mathcal{T}W)(s)| &\leq \max_{a,b} |M_s(V)_{ab} - M_s(W)_{ab}| \\ &\leq \gamma \max_{a,b} \sum_{s'} P(s' \mid s, a, b) |V(s') - W(s')| \\ &\leq \gamma \|V - W\|_\infty. \end{aligned}$$

Hence $\|\mathcal{T}V - \mathcal{T}W\|_\infty \leq \gamma \|V - W\|_\infty$.

2-player zero-sum Markov games

$$V_{k+1} = \mathcal{T}V_k, \quad \|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty.$$

- Each update solves one zero-sum matrix game per state.
- With $V_0 = 0$, value error is at most $\gamma^k/(1 - \gamma)$.
- Thus the standard iteration bound **depends on** $1/(1 - \gamma)$, not just its number of input bits.

Convergence alone **does not establish polynomial running time** in the binary input size.

Policy Gradient Iteration

For each player k , state s and action a , let

$$\pi_k(a \mid s) = x_{k,s,a}, \quad x_{k,s,a} \geq 0, \quad \sum_{a \in \mathcal{A}_k} x_{k,s,a} = 1.$$

The policy vector x_k lies in a **product of simplices**: $\Delta(\mathcal{A}_k)^{|S|}$.

Policy Gradient Iteration

Each player moves in the direction that **increases her own value**, then projects back to valid policies:

$$x_k^{(t+1)} = \Pi_{\Delta(\mathcal{A}_k)|\mathcal{S}|} \left(x_k^{(t)} + \eta_k \nabla_{x_k} V_k^{x^{(t)}}(\rho) \right).$$

Π is Euclidean projection. The step size is $\eta_k > 0$.

In a zero-sum game, ascent on both players' own rewards is descent-ascent on the common payoff V .

Some facts about Policy Gradient

One agent. With exact gradients, a suitable step size and adequate state coverage, **projected policy gradient** reaches an ε -optimal policy in $O(\varepsilon^{-2})$ iterations. *Agarwal et al. (2020/2021).*

Two-player zero-sum. **Two-timescale policy-gradient** methods give polynomial approximation guarantees under exploration and coverage assumptions. Daskalakis, Foster and Golowich (2020).

These are **not general convergence guarantees** for all games. Even approximate stationary Markov CCE is **PPAD-hard** in two-player general-sum games.

Daskalakis, Golowich and Zhang (2023).