INFO8003-1
Reinforcement Learning

Multi-Agent Reinforcement Learning

Hansen Julien (julien.hansen@uliege.be)

April 2026

Content

Content

Today, an overview of multi-agent reinforcement learning (MARL):

- Reminder: Single-Agent RL
- Why Multi-Agent RL?
- Extension to Multi-Agent RL
- Hierarchy and Classes of Games
- Solution Concepts for Games
- Paradigms in MARL
- Challenges Inherent to MARL
- Current Methods
- Applications
- References

Reminder: Single-Agent RL

Markov Decision Process (MDP)

A **Markov Decision Process** is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$:

- $\mathcal{S}$: state space
- $\mathcal{A}$: action space
- $T(s'|s, a)$: transition function
- $R(s, a, s')$: reward function
- $\gamma \in [0, 1)$: discount factor

Goal: Find a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximises the expected discounted return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \, R(s_t, a_t, s_{t+1}) \right]$$

Value Functions

State-value function:

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

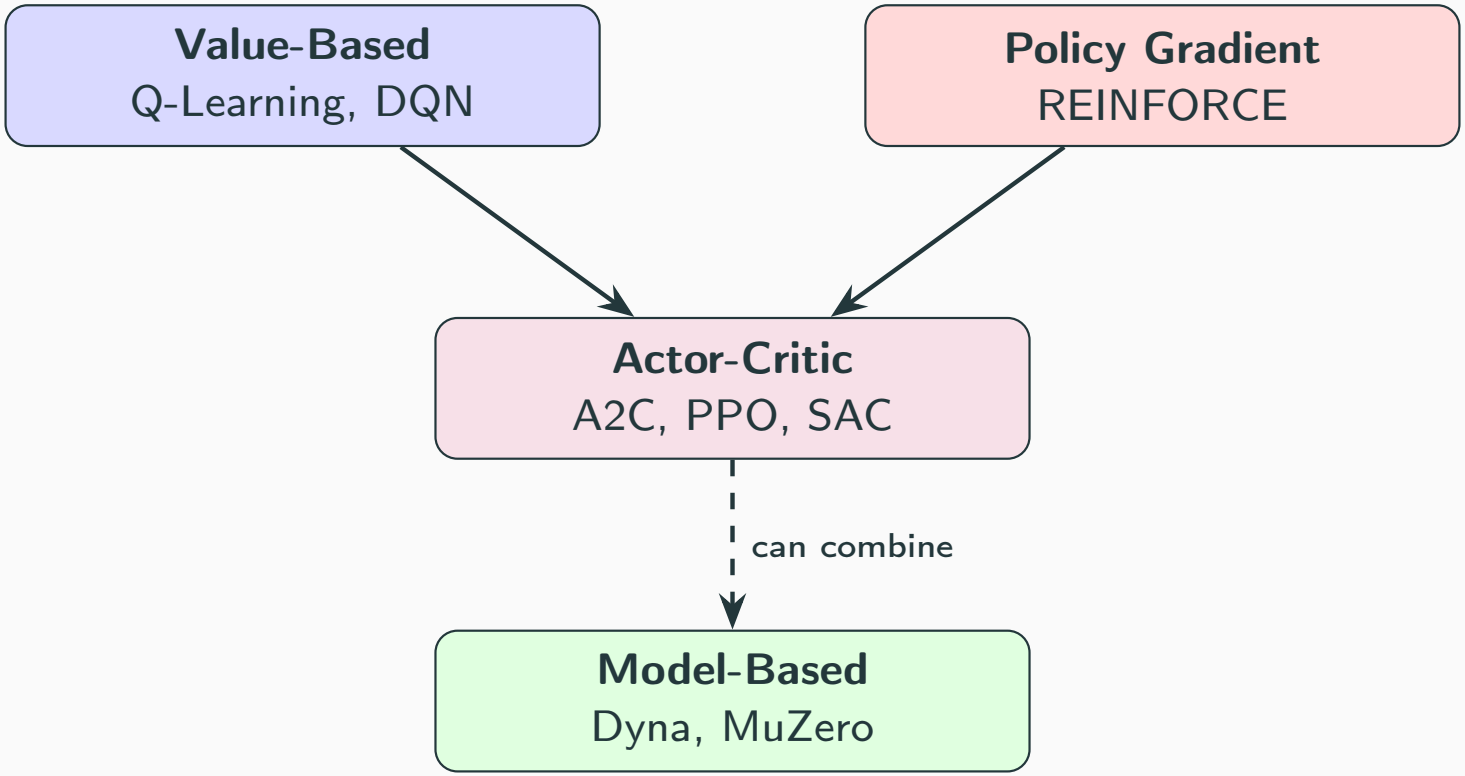
Action-value function:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} r_t \,\middle|\, s_0 = s,\ a_0 = a\right]$$

Bellman optimality equation:

$$Q^{*}(s, a) = \sum_{s'} T(s'|s, a)\left[R(s, a, s') + \gamma \max_{a'} Q^{*}(s', a')\right]$$

Classes of RL Algorithms



Value-Based Methods

Q-Learning (off-policy, tabular):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Deep Q-Network (DQN): approximate Q^* with a neural network Q_θ

- Experience replay
- Target network for stability
- Extensions: Double DQN, Dueling DQN, Rainbow

Policy Gradient Methods

Parameterise the policy π_θ directly.

Policy Gradient Theorem (Sutton et al., 1999):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\ Q^{\pi_\theta}(s,a)]$$

Key algorithms:

- REINFORCE (Monte-Carlo)
- Advantage Actor-Critic (A2C / A3C)
- PPO, clipped surrogate objective
- SAC, maximum entropy framework

Single-Agent RL: Summary

Family	Learns	Examples
Value-Based	Q^*	DQN, Rainbow
Policy Gradient	π_θ	REINFORCE, PPO
Actor-Critic	Both	A2C, SAC
Model-Based	T, R	Dyna, MuZero

Fundamental assumption: the environment is *stationary*.

A single agent interacts with a fixed MDP.

What happens when *other learning agents* are part of the environment?

Why Multi-Agent RL?

Limitations of Single-Agent RL

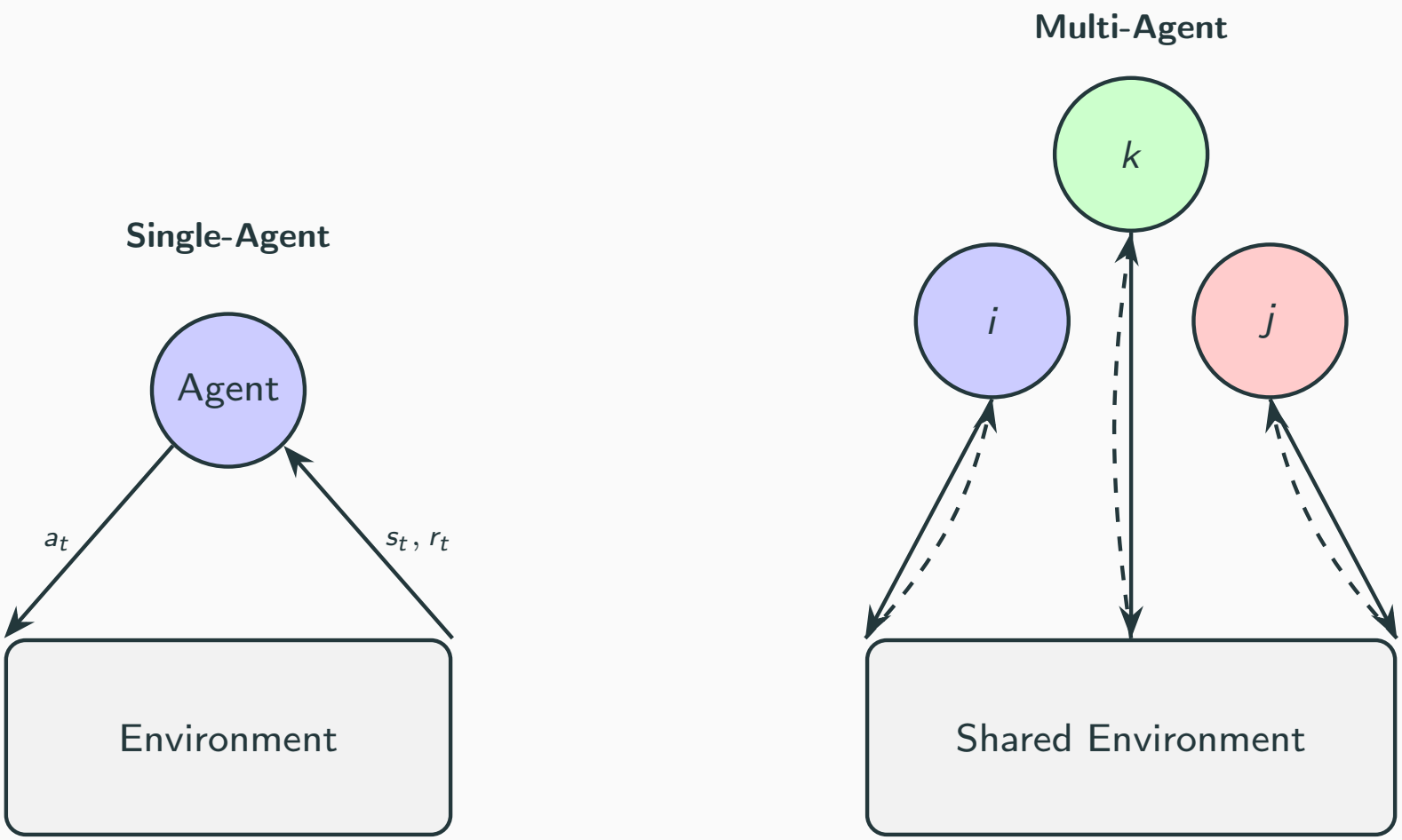
Many real-world problems are inherently **multi-agent**:

- Autonomous driving
- Robotic teams
- Financial markets
- Game AI

Modelling other agents as part of the environment is a **strong simplification**:

- Other agents *learn* and *adapt* $\Rightarrow$ environment is **non-stationary**
- Convergence guarantees of single-agent RL **break down**

From Single to Multi-Agent



Each agent i selects action a_t^i; the joint action $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^n)$ determines the next state and individual rewards.

Motivating Examples

Cooperative:

- Robot swarm coordination
- Multi-robot warehouse (e.g. Amazon Robotics)
- Cooperative card games (e.g. Hanabi)

Competitive:

- Board games (Go, Chess, Poker)
- Adversarial security (attacker–defender)
- Sports strategy (e.g. Google Research Football)

Mixed (Cooperative–Competitive):

- Autonomous traffic
- Economic markets
- Hide-and-seek (OpenAI, 2019)

Extension to Multi-Agent RL

From MDP to Stochastic Game

A **Stochastic Game** (Shapley, 1953) extends the MDP to n agents:

$$\mathcal{G} = \langle \mathcal{N},\ \mathcal{S},\ \{\mathcal{A}^i\}_{i\in\mathcal{N}},\ T,\ \{R^i\}_{i\in\mathcal{N}},\ \gamma \rangle$$

- $\mathcal{N} = \{1, \ldots, n\}$: set of agents
- $\mathcal{S}$: global state space
- $\mathcal{A}^i$: action space of agent i; $\boldsymbol{\mathcal{A}} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$
- $T(s'|s, \boldsymbol{a})$: joint transition
- $R^i(s, \boldsymbol{a}, s')$: reward for agent i
- γ: discount factor

Note: when $n = 1$, this reduces to an MDP.

Partially Observable Stochastic Game (POSG)

In many settings, agents have **partial observability**:

$$\mathcal{G}_{\text{PO}} = \langle \mathcal{N},\ \mathcal{S},\ \{\mathcal{A}^i\},\ T,\ \{R^i\},\ \{\mathcal{O}^i\},\ \{O^i\},\ \gamma \rangle$$

Additional components:

- $\mathcal{O}^i$: observation space of agent i
- $O^i(o^i|s, \boldsymbol{a})$: observation function

Each agent i maintains a **history**:

$$\tau_t^i = (o_0^i, a_0^i, o_1^i, a_1^i, \ldots, o_t^i)$$

and a policy $\pi^i(a^i|\tau_t^i)$ conditioned on its own history.

Decentralised POMDP (Dec-POMDP)

A **Dec-POMDP** (Bernstein et al., 2002) is a special case of a POSG where all agents share a **common reward**:

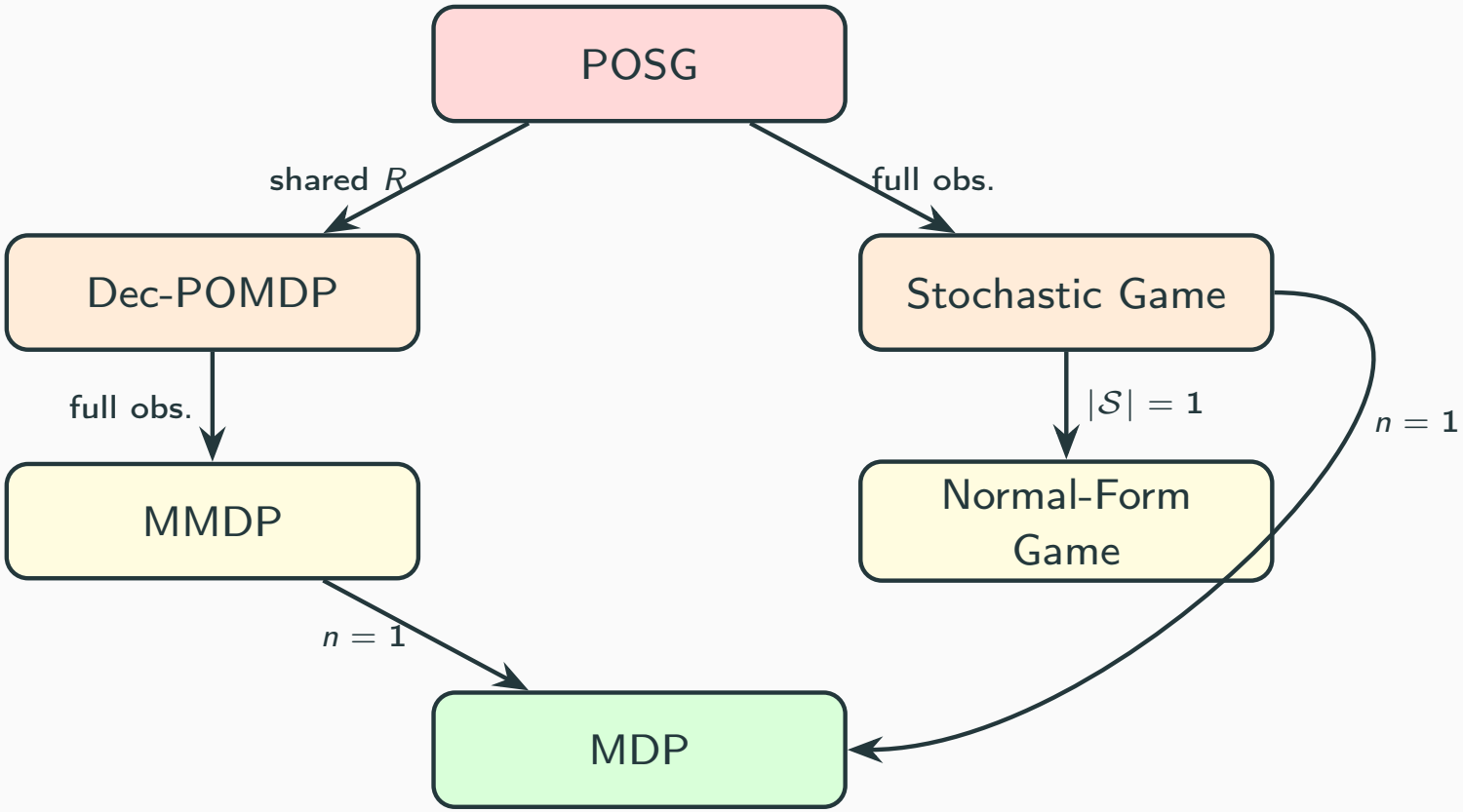
$$R^1 = R^2 = \cdots = R^n = R$$

- Fully **cooperative** setting
- Agents must coordinate under partial observability

Other notable frameworks:

- **MMDP**: multi-agent MDP (full observability, shared reward)
- **POSG**: most general (individual rewards, partial obs.)
- **Normal-form game**: single-state stochastic game

Formalism Hierarchy



Hierarchy and Classes of Games

Classifying Multi-Agent Interactions

Games can be classified along several axes:

1. **Reward structure:** cooperative, competitive, mixed
2. **Observability:** full vs. partial
3. **Number of agents:** 2-player vs. many-player
4. **Action timing:** simultaneous vs. sequential
5. **Horizon:** one-shot (normal-form) vs. repeated / sequential

Cooperative Games

All agents share a **common objective**:

$$R^1(s, \boldsymbol{a}, s') = R^2(s, \boldsymbol{a}, s') = \cdots = R^n(s, \boldsymbol{a}, s')$$

Challenges:

- Credit assignment, which agent caused the reward?
- Coordination, multiple equivalent optima
- Communication, agents may need to share information

Examples: multi-robot coordination, Hanabi, StarCraft micromanagement

Competitive Games

Agents have **opposing objectives**.

In two-player **zero-sum** games:

$$R^1(s, \boldsymbol{a}, s') = -R^2(s, \boldsymbol{a}, s')$$

Key result: Minimax theorem (von Neumann, 1928):

$$\max_{\pi^{\mathbf{1}}} \min_{\pi^{\mathbf{2}}} J^1(\pi^1, \pi^2) = \min_{\pi^{\mathbf{2}}} \max_{\pi^{\mathbf{1}}} J^1(\pi^1, \pi^2)$$

Implications:

- A well-defined *value* of the game exists
- Minimax strategies form a Nash equilibrium

Examples: Go (AlphaGo), Chess, Poker (Libratus, Pluribus)

Mixed-Motive (General-Sum) Games

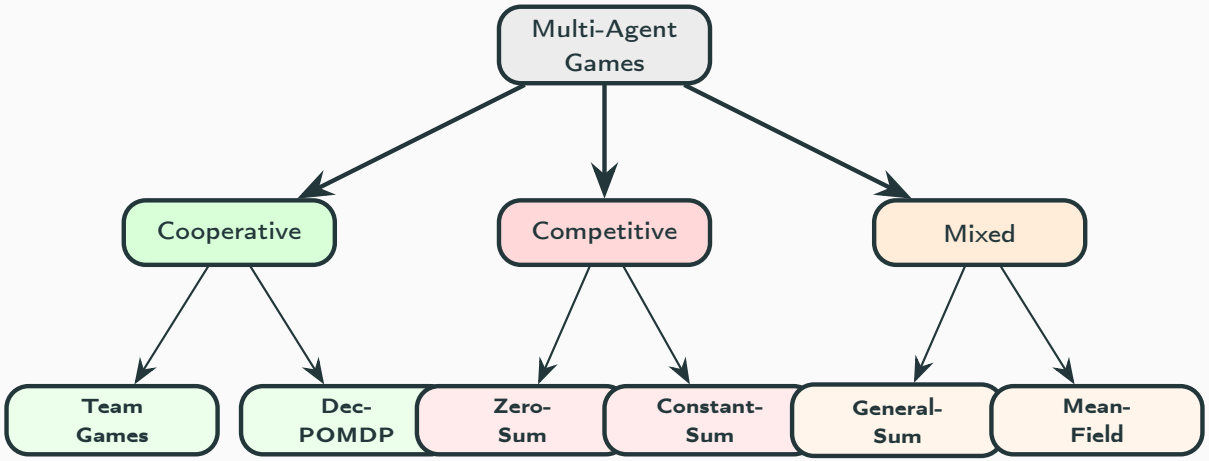
Agents have **individual** reward functions with no fixed relationship.

Classic example: Prisoner's Dilemma:

	Cooperate	**Defect**
Cooperate	$(-1, -1)$	$(-3,\ 0)$
Defect	$(0, -3)$	$(-2, -2)$

- Individual rationality leads to *mutual defection*
- Social optimum requires *cooperation*
- Tension between individual and collective goals

Taxonomy Overview



Solution Concepts for Games

What Does "Solving" a Game Mean?

In single-agent RL, the solution is an **optimal policy** π^*.

In multi-agent settings, optimality depends on *all agents' policies*:

$$J^i(\pi^i, \boldsymbol{\pi}^{-i}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, R^i(s_t, \boldsymbol{a}_t, s_{t+1}) \;\middle|\; \pi^i, \boldsymbol{\pi}^{-i}\right]$$

where $\boldsymbol{\pi}^{-i}$ denotes the policies of all agents except i.

We need **solution concepts** from game theory to define what a "good" outcome is.

Joint Policy

A **joint policy** is the collection of all agents' policies:

$$\boldsymbol{\pi} = (\pi^1, \pi^2, \ldots, \pi^n)$$

The environment dynamics under a joint policy become:

$$\Pr(s_{t+1}|s_t, \boldsymbol{\pi}) = \sum_{\boldsymbol{a}} \left(\prod_{i=1}^{n} \pi^i(a^i|s_t) \right) T(s_{t+1}|s_t, \boldsymbol{a})$$

Each agent's return depends on the **entire joint policy**, not just its own.

Best Response

The **best response** of agent i given the other agents' policies $\boldsymbol{\pi}^{-i}$:

$$BR^i(\boldsymbol{\pi}^{-i}) = \arg\max_{\pi^i} J^i(\pi^i, \boldsymbol{\pi}^{-i})$$

Intuition: fix everyone else's strategy, then find the single-agent optimal policy.

Key insight

Computing a best response reduces to solving a (single-agent) MDP, since the other policies are fixed $\Rightarrow$ the environment is stationary from agent i's perspective.

Nash Equilibrium

A joint policy $\boldsymbol{\pi}^*$ is a **Nash Equilibrium** (NE) if no agent can unilaterally improve:

$$\forall\, i,\quad J^i(\pi^{i*}, \boldsymbol{\pi}^{-i*}) \geq J^i(\pi^i, \boldsymbol{\pi}^{-i*}),\quad \forall\, \pi^i$$

Equivalently, every agent plays a best response:

$$\pi^{i*} \in BR^i(\boldsymbol{\pi}^{-i*}),\quad \forall\, i \in \mathcal{N}$$

Properties:

- Every finite game has at least one NE (possibly in mixed strategies)
- NE may not be unique or Pareto-optimal
- Finding NE is **PPAD-complete** in general

ε-Nash Equilibrium

In practice, we often settle for approximate solutions.

A joint policy $\boldsymbol{\pi}$ is an **ε-Nash equilibrium** if:

$$\forall\, i, \quad J^i(\pi^i, \boldsymbol{\pi}^{-i}) \geq \max_{\hat{\pi}^i} J^i(\hat{\pi}^i, \boldsymbol{\pi}^{-i}) - \varepsilon$$

Exploitability measures the distance to a NE:

$$\text{Exploit}(\boldsymbol{\pi}) = \sum_{i=1}^{n} \left[\max_{\hat{\pi}^i} J^i(\hat{\pi}^i, \boldsymbol{\pi}^{-i}) - J^i(\pi^i, \boldsymbol{\pi}^{-i}) \right]$$

At a NE, exploitability is zero.

Other Solution Concepts

- **Correlated Equilibrium (CE):** agents follow recommendations from a shared signal; generalises NE; can be computed in polynomial time via LP
- **Pareto Optimality:** no agent can improve without making another worse off; relevant in cooperative / social welfare settings
- **Minimax:** in zero-sum games, equivalent to NE; agent maximises its worst-case payoff
- **Stackelberg Equilibrium:** one agent (leader) commits first, the other (follower) best-responds

Paradigms in MARL

Three Training Paradigms

The key question: **What information is available during training vs. execution?**

Fully Centralised

Idea: a single controller observes *everything* and commands all agents.

Training: global state + all actions $\Rightarrow$ learn a joint policy $\pi(\boldsymbol{a}|s)$

Execution: central controller dispatches actions

Pros:

- Reduces to single-agent RL
- No non-stationarity

Cons:

- Joint action space grows *exponentially*: $|\boldsymbol{\mathcal{A}}| = \prod_i |\mathcal{A}^i|$
- Requires a central controller at execution (single point of failure)
- Not applicable when agents have private observations

Fully Decentralised

Idea: each agent learns independently using only its own experience.

Training: each agent treats others as part of the environment

Execution: each agent acts on its own observations

Pros:

- Simple, scalable
- No communication needed

Cons:

- **Non-stationarity:** other agents' policies change during training
- Credit assignment is difficult
- May fail to coordinate

Example: Independent Q-Learning (IQL) — each agent runs its own DQN.

Centralised Training, Decentralised Execution (CTDE)

Idea: exploit global information during *training*, but execute *decentrally*.

Training: access to global state, all observations, joint actions

Execution: each agent uses only its local observation o^i

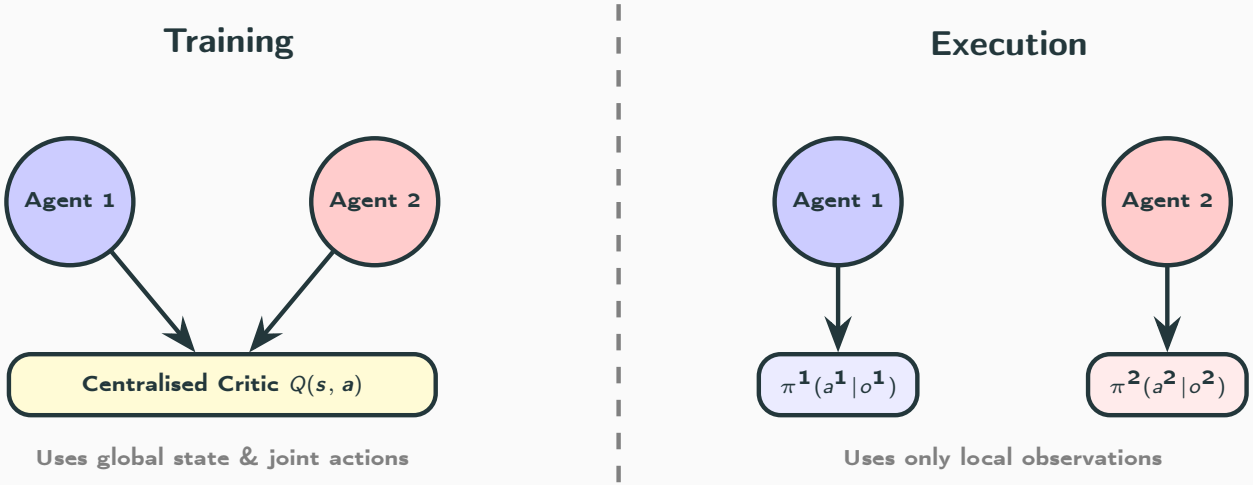
Why it works:

- Training can use extra information to stabilise learning
- Execution policies only depend on local information
- Practical for real-world deployment

Key methods:

- QMIX, VDN
- MADDPG, MAPPO
- COMA

CTDE: Illustration



Challenges Inherent to MARL

Challenge 1: Non-Stationarity

From agent i's perspective, the environment includes all other agents.

As other agents learn and change their policies:

$$T^i_{\text{eff}}(s'|s, a^i) = \sum_{\boldsymbol{a}^{-i}} \left(\prod_{j \neq i} \pi^j_t(a^j|s) \right) T(s'|s, a^i, \boldsymbol{a}^{-i})$$

T^i_{eff} changes over time as $\{\pi^j_t\}_{j \neq i}$ evolve.

Consequence

Convergence guarantees of Q-Learning, policy gradient, etc. rely on a **stationary** MDP assumption. This assumption is **violated** in MARL.

Challenge 2: Scalability

The joint action space grows **exponentially** with the number of agents:

$$|\boldsymbol{\mathcal{A}}| = \prod_{i=1}^{n} |\mathcal{A}^i|$$

Example: 10 agents, 5 actions each $\Rightarrow 5^{10} \approx 10^7$ joint actions.

Mitigation strategies:

- Value decomposition (QMIX, QTRAN)
- Mean-field approximations
- Parameter sharing across agents
- Attention mechanisms for agent communication

Challenge 3: Credit Assignment

When agents receive a **shared team reward**, how to determine each agent's contribution?

The credit assignment problem:

- Agent i takes action a_t^i
- Team receives reward r_t
- Was agent i's action good or bad?

Approaches:

- **Difference rewards:** $r^i = r - r(\text{without agent } i)$
- **Counterfactual baselines** (COMA): compare to default action
- **Shapley values:** marginal contribution across coalitions
- **Value decomposition:** learn individual Q^i from joint Q

Challenge 4: Partial Observability

Agents typically observe only a **local view** of the environment:

- Limited sensor range
- Communication constraints
- Information asymmetry

Consequences:

- Agents must reason about hidden state and other agents' beliefs
- Optimal policies may require *memory* (history-dependent)
- Communication becomes a learnable action

Approaches:

- Recurrent policies (GRU/LSTM over observation history)
- Learned communication channels (CommNet, TarMAC)
- Theory of Mind models

Challenge 5: Equilibrium Selection

Even when a Nash equilibrium exists, there may be **multiple equilibria**:

- Which one should agents converge to?
- Coordination failure: agents may converge to *different* equilibria

Coordination game example:

	Left	**Right**
Left	(2, 2)	(0, 0)
Right	(0, 0)	(1, 1)

Two NE: (Left, Left) and (Right, Right). How do independent learners coordinate?

Mitigations: conventions, communication protocols, population-based training.

Current Methods

Independent Learners

Independent Q-Learning (IQL):

- Each agent i maintains its own $Q^i(o^i, a^i)$
- Treats other agents as part of the environment
- Simple but no convergence guarantees (non-stationarity)

Independent PPO (IPPO):

- Each agent runs PPO independently
- Surprisingly effective in many cooperative tasks (Yu et al., 2022)
- Often used as a strong baseline

Insight

Despite theoretical limitations, independent learners with parameter sharing can be remarkably competitive.

Value Decomposition: VDN & QMIX

Goal: decompose the joint action-value $Q_{\text{tot}}(\boldsymbol{s}, \boldsymbol{a})$ into individual utilities.

VDN (Sunehag et al., 2018): additive decomposition

$$Q_{\text{tot}}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{n} Q^i(\tau^i, a^i)$$

QMIX (Rashid et al., 2018): monotonic mixing

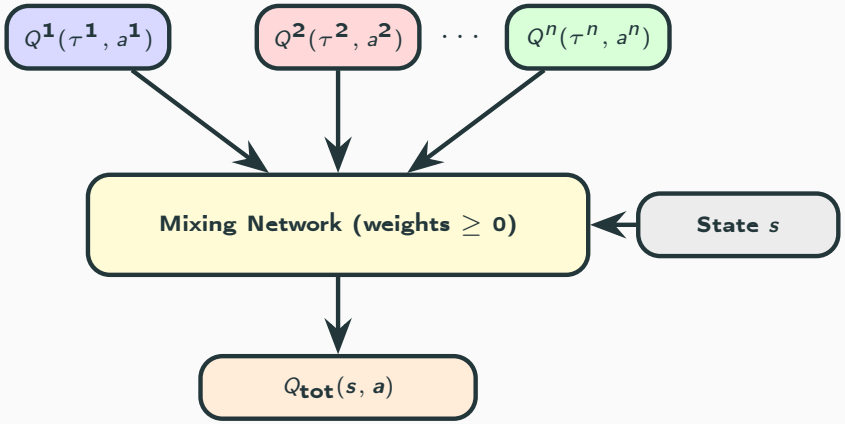
$$Q_{\text{tot}}(\boldsymbol{s}, \boldsymbol{a}) = f_{\text{mix}}\big(Q^1(\tau^1, a^1), \ldots, Q^n(\tau^n, a^n); \boldsymbol{s}\big)$$

with constraint:

$$\frac{\partial Q_{\text{tot}}}{\partial Q^i} \geq 0, \quad \forall\, i$$

Ensures: $\arg\max_{\boldsymbol{a}} Q_{\text{tot}} = \big(\arg\max_{a^1} Q^1, \ldots, \arg\max_{a^n} Q^n\big)$

QMIX Architecture



- Hypernetwork generates mixing weights conditioned on state $\boldsymbol{s}$
- Weights constrained to be non-negative (monotonicity)
- Allows decentralised execution via individual arg max

QMIX: The IGM Principle

Individual-Global-Max (IGM) — the key property that enables decentralised execution:

$$\arg\max_{\boldsymbol{a}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a^1} Q^1(\tau^1, a^1) \\ \vdots \\ \arg\max_{a^n} Q^n(\tau^n, a^n) \end{pmatrix}$$

QMIX enforces IGM via monotonicity:

$$\frac{\partial Q_{\text{tot}}}{\partial Q^i} \geq 0 \quad \forall\, i \quad \Longrightarrow \quad \text{IGM holds}$$

Training loss (end-to-end, from joint replay buffer):

$$\mathcal{L}(\theta) = \mathbb{E}_{(\boldsymbol{\tau}, \boldsymbol{a}, r, \boldsymbol{\tau}') \sim \mathcal{D}} \left[\left(r + \gamma\, Q^{\bar{\theta}}_{\text{tot}}(\boldsymbol{\tau}', \boldsymbol{a}'^{*}) - Q^{\theta}_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) \right)^2 \right]$$

where $\boldsymbol{a}'^{*} = (\arg\max_{a'^1} Q^1_{\bar{\theta}}(\tau'^1, a'^1), \dots)$ and $\bar{\theta}$ is a target network.

QMIX: Limitations

The **monotonicity constraint** limits representational capacity.

Example: Payoff matrix that QMIX **cannot** represent:

	$a^2 = A$	$a^2 = B$
$a^1 = A$	8	-12
$a^1 = B$	-12	0

Optimal joint action: (A, A) with value 8.
But $Q^1(A)$ and $Q^2(A)$ must both be highest (IGM), while row/column averages suggest B is safer $\Rightarrow$ monotonicity is too restrictive.

This motivates QTRAN, QPLEX, WQMIX and other methods that relax or remove the monotonicity constraint.

Multi-Agent Policy Gradient: MADDPG

MADDPG (Lowe et al., 2017): Multi-Agent DDPG

Key idea: centralised critic + decentralised actors

- Each agent i has:
  - Actor: $\mu^i(o^i)$ — depends only on local observation
  - Critic: $Q^i(\boldsymbol{x}, a^1, \ldots, a^n)$ — uses all agents' info
- Critic trained with:

$$\mathcal{L}(\theta^i) = \mathbb{E}\left[\left(Q^i_{\theta^i}(\boldsymbol{x}, \boldsymbol{a}) - y\right)^2\right]$$

where $y = r^i + \gamma\, Q^i_{\bar{\theta}^i}(\boldsymbol{x}', \mu^1(o'^1), \ldots, \mu^n(o'^n))$

Works in cooperative, competitive, and mixed settings.

From PPO to MAPPO: Recap of PPO

Single-agent PPO — clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t,\ \ \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},\, 1-\epsilon,\, 1+\epsilon\right)\hat{A}_t\right)\right]$$

Three components in PPO:

1. **Policy** $\pi_\theta(a|s)$: maps state to action distribution
2. **Value function** $V_\phi(s)$: estimates expected return
3. **Advantage** $\hat{A}_t = \sum_{l=0}^{T-t}(\gamma\lambda)^l\delta_{t+l}$ via GAE, where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$

Question: What changes when we have n agents?

From PPO to MAPPO: The Multi-Agent Extension

Three modifications to go from PPO to MAPPO:

1. **Decentralise the policy** — each agent i has its own:

$$\pi_\theta^i(a_t^i \mid o_t^i) \qquad \text{(local observation only)}$$

2. **Centralise the value function** — uses global state:

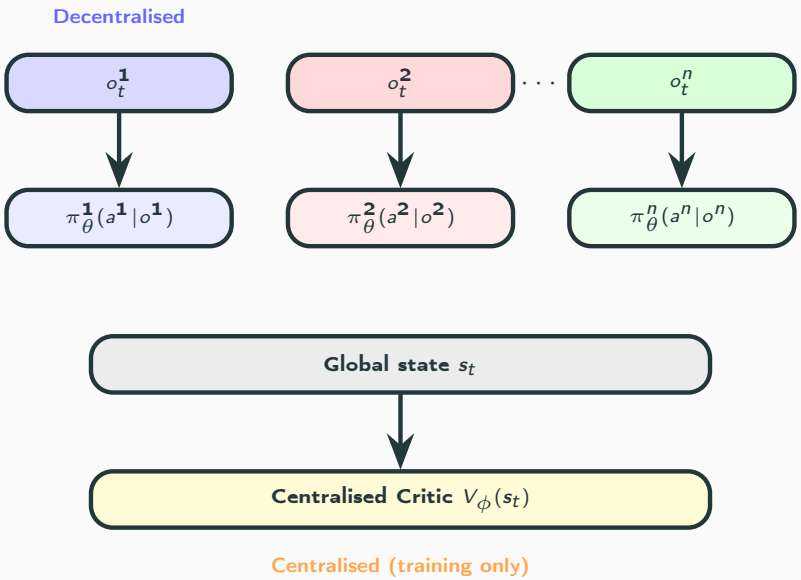
$$V_\phi(\boldsymbol{s}_t) \quad \text{or} \quad V_\phi^i(o_t^i, \boldsymbol{s}_t)$$

3. **Compute per-agent advantages:**

$$\hat{A}_t^i = \sum_{l=0}^{T-t} (\gamma\lambda)^l \left(r_t^i + \gamma\, V_\phi(\boldsymbol{s}_{t+l+1}) - V_\phi(\boldsymbol{s}_{t+l}) \right)$$

Each agent's clipped objective uses its *own* ratio $\frac{\pi_\theta^i}{\pi_{\theta_{\text{old}}}^i}$ and advantage $\hat{A}_t^i$.

MAPPO: Architecture & Practical Details



Implementation details (Yu et al., 2022):

- **Parameter sharing:** all agents use the same policy network (+ agent ID as input)
- Achieves SOTA on SMAC, MPE, Hanabi, and Google Research Football

Counterfactual Multi-Agent Policy Gradient (COMA)

COMA (Foerster et al., 2018): addresses credit assignment via counterfactual baselines.

Advantage for agent i:

$$A^i(s, \boldsymbol{a}) = Q(s, \boldsymbol{a}) - \sum_{a'^i} \pi^i(a'^i|\tau^i)\, Q(s, (a'^i, \boldsymbol{a}^{-i}))$$

Intuition:

- Compare the joint Q-value of the action *actually taken*...
- ...to the expected Q-value under agent i's current policy
- Isolates agent i's individual contribution

Requires a centralised critic that can evaluate counterfactual actions.

Communication Learning

Some methods learn to **communicate** between agents:

CommNet (Sukhbaatar et al., 2016):

- Continuous communication channel
- Average message passing across agents

TarMAC (Das et al., 2019):

- Targeted multi-agent communication
- Soft attention to select which agents to communicate with

DIAL (Foerster et al., 2016):

- Differentiable inter-agent learning
- Messages are discrete during execution, continuous during training

Self-Play Methods

For **competitive** settings — train by playing against yourself:

Naïve Self-Play:

- Train against the *current* copy of yourself
- AlphaGo / AlphaZero: MCTS + self-play
- Risk: beats current self but forgets how to beat earlier versions

Fictitious Self-Play (FSP):

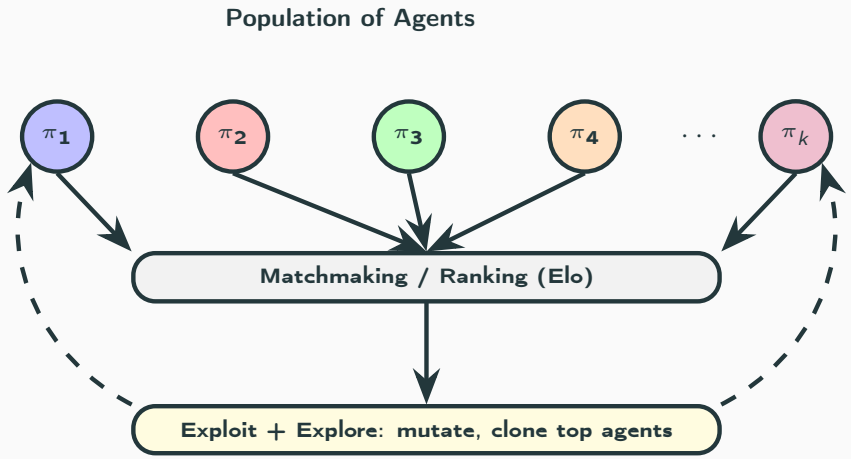
- Best-respond to the *average* of all past opponent policies

Limitation

Both methods produce a *single* agent. What if we want *robust* strategies that handle diverse opponents?

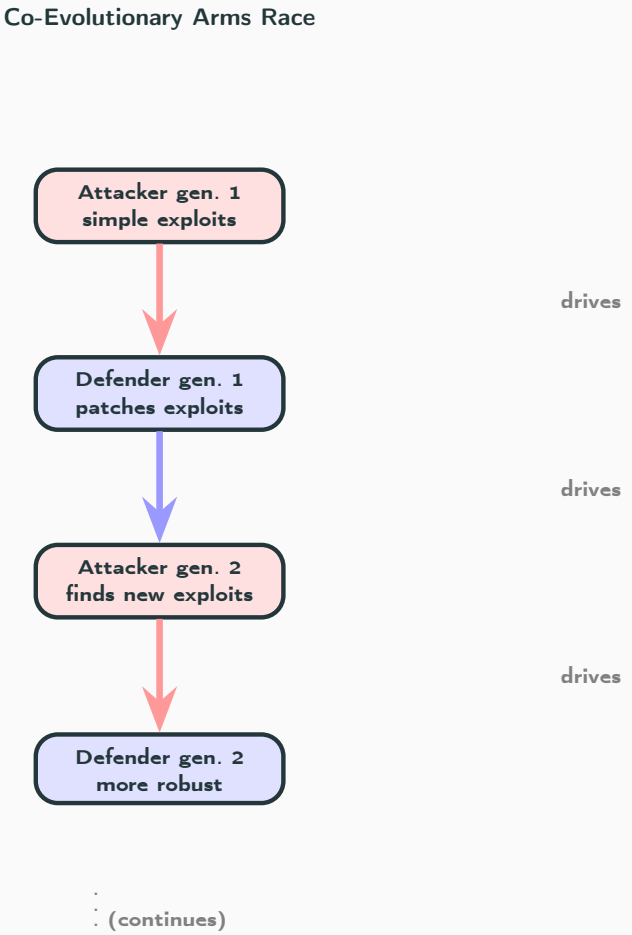
Population-Based Training (PBT)

Idea: maintain a **diverse population** of agents that co-evolve.



- Agents train against *diverse* opponents $\Rightarrow$ robust strategies
- Low-performing agents are replaced by mutated copies of top agents
- **AlphaStar**: structured league (Main / Exploiter / League Exploiter)
- **OpenAI Five**: population-based hyperparameter tuning

PBT, Adversarial Robustness & Co-Evolution



Key insight: population diversity $\Rightarrow$ adversarial robustness.

Applications:

- **Game AI**: AlphaStar league prevents over-specialisation against one play-style
- **LLM red-teaming**: diverse population of attackers finds more vulnerabilities
- **Cybersecurity**: attacker–defender co-evolution
- **Robust control**: train against population of disturbances

Connection to Hide-and-Seek:

- Each counter-strategy creates selection pressure for the next

Opponent Modelling

Idea: explicitly model other agents' policies and reason about them.

Level 0: Ignore others (Independent Learners)
Level 1: Model others as stationary $\hat{\pi}^{-i}$, best-respond
Level 2: Model others as *learning* agents, anticipate adaptation

Approaches:

- Bayesian opponent modelling: maintain belief over opponent type
- Neural opponent models: predict π^{-i} from observation history
- Theory of Mind: recursive reasoning about beliefs

Key question: Can we go further and *shape* the opponent's learning?

LOLA: Learning with Opponent-Learning Awareness

LOLA (Foerster et al., 2018b): differentiate *through* the opponent's learning step.

Standard gradient:

$$\Delta\theta^1 \propto \nabla_{\theta^1} J^1(\theta^1, \theta^2)$$

LOLA gradient: account for opponent's next update $\Delta\theta^2$:

$$\Delta\theta^1 \propto \nabla_{\theta^1} J^1(\theta^1, \theta^2 + \Delta\theta^2) \approx \underbrace{\nabla_{\theta^1} J^1}_{\text{naïve}} + \underbrace{\nabla_{\theta^1}\theta^2 \cdot \nabla_{\theta^2} J^1}_{\text{opponent shaping}}$$

Result: In iterated Prisoner's Dilemma, LOLA agents learn to **cooperate**, while naïve learners converge to mutual defection.

Extensions: DiCE (higher-order), M-FOS (meta-learned opponent shaping)

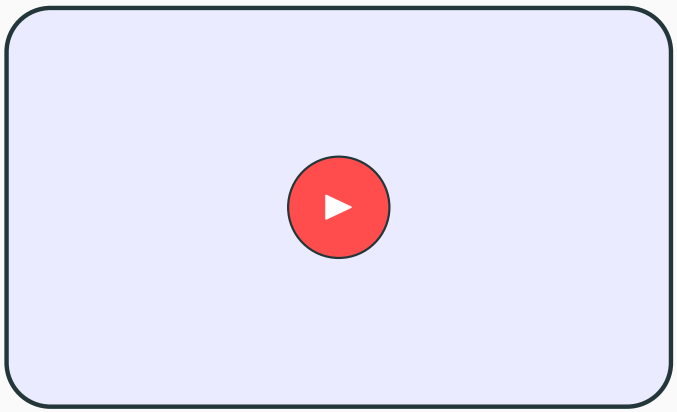
Methods Summary

Method	Paradigm	Setting	Key Idea
IQL / IPPO	Decentralised	Any	Independent learners
VDN	CTDE	Cooperative	Additive decomposition
QMIX	CTDE	Cooperative	Monotonic mixing
QTRAN	CTDE	Cooperative	Unrestricted decomposition
MADDPG	CTDE	Any	Centralised critic (DDPG)
MAPPO	CTDE	Cooperative	Centralised critic (PPO)
COMA	CTDE	Cooperative	Counterfactual baseline
CommNet	CTDE	Cooperative	Learned communication
Self-Play	Centralised	Competitive	Train vs. self
PBT / League	Population	Competitive	Diverse co-evolution
LOLA	Decentralised	Any	Opponent-learning awareness

State-of-the-Art Applications

AlphaStar — StarCraft II (DeepMind, 2019)

DeepMind official presentation



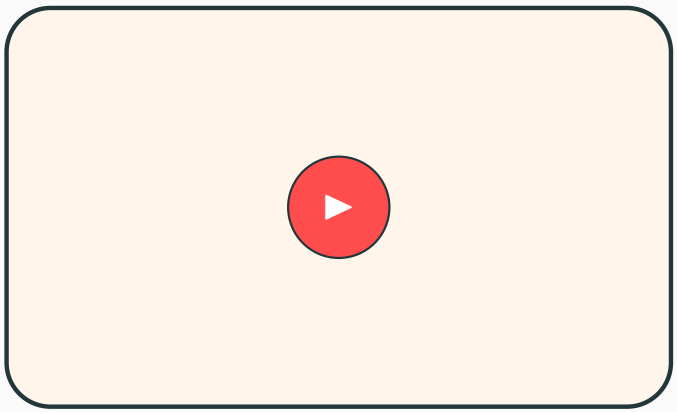
Click to watch on YouTube

- **Grandmaster** level in StarCraft II (top 0.2% of players)
- **League training**: Main Agents, Main Exploiters, League Exploiters + frozen past policies
- Multi-agent: competitive (opponent) + cooperative (unit micro-management)
- Partial observability (fog of war)
- Each race: single neural network

Video: youtu.be/cUTMhmVh1qs

OpenAI Five — Dota 2 (OpenAI, 2019)

OpenAI Five Finals vs OG



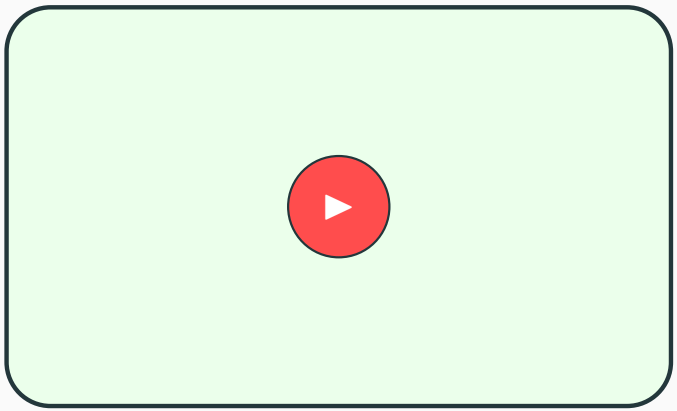
Click to watch on YouTube

- 5v5 team game
- **Independent PPO** with shared team reward
- Each hero: separate 1024-unit LSTM
- 128k CPUs, 256 GPUs; 180 years/day of gameplay
- Beat world champion team OG (2019)
- Key insight: simple algorithms + massive scale

Video: youtu.be/tfb6aEUMC04

Pluribus — Multiplayer Poker (Meta, 2019)

Two Minute Papers summary



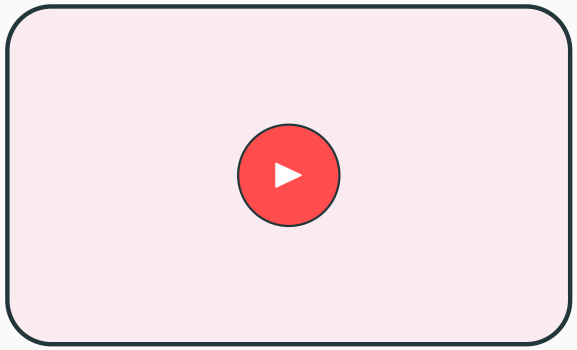
Click to watch on YouTube

- First superhuman AI for **6-player** No-Limit Texas Hold'em
- **Two phases:**
  - Offline: blueprint via fictitious self-play
  - Online: real-time search at decision points
- Imperfect information: bluffing, deception
- Ran on a **single server** ($150 of compute)
- Algorithmic innovation > brute-force scale

Video: https://www.youtube.com/watch?v=u90TbxK7VEA

Emergent Tool Use — Hide-and-Seek (OpenAI, 2020)

OpenAI official video

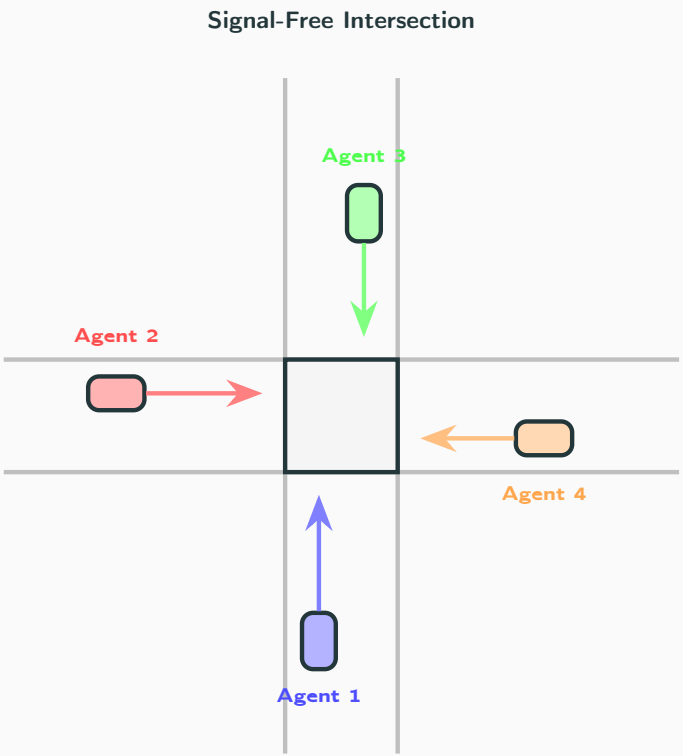


Click to watch on YouTube

- 2 hiders vs. 2 seekers
- 6 emergent phases:
  1. Random running
  2. Chase / flee
  3. Hiders build shelters
  4. Seekers use ramps to climb walls
  5. Hiders lock ramps
  6. Seekers "box surf" $\rightarrow$ hiders lock everything
- **Autocurriculum:** no reward shaping needed

Video: youtu.be/kopoLzvh5jY
Also: Two Minute Papers summary

Autonomous Driving



Mixed-motive MARL:

- Cooperate: avoid collisions
- Compete: reach destination fast

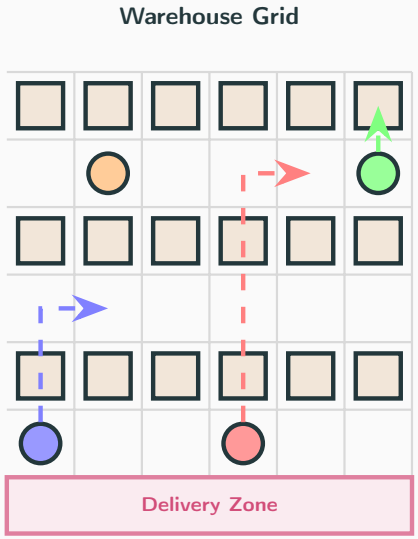
Applications:

- Signal-free intersection management
- Highway merging & lane changing
- Traffic signal control across intersections (PressLight, CoLight)

Key challenges:

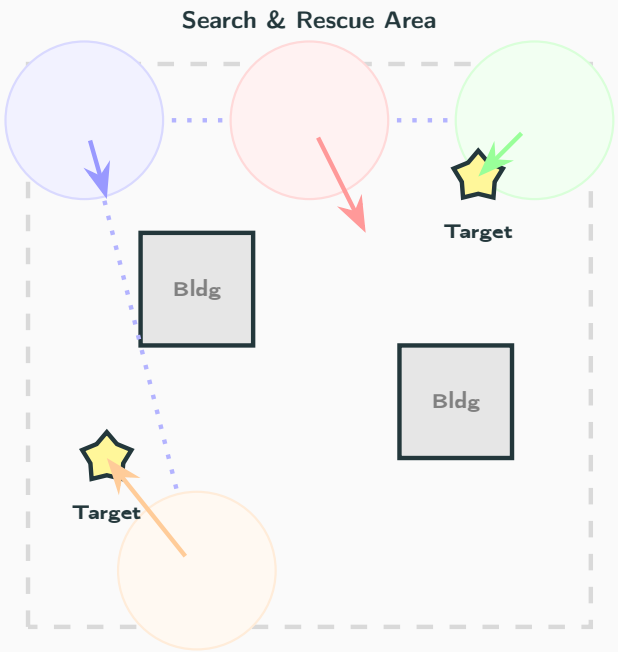
- Hard safety constraints
- Heterogeneous agents
- Sim-to-real transfer

Multi-Robot Warehouse



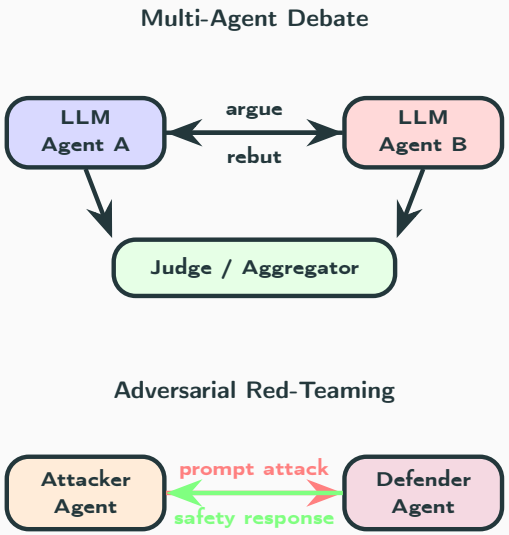
- Robots must pick items and deliver to shipping area
- **Dec-POMDP**: each robot sees only nearby area
- **Challenges:**
  - Path planning without collisions
  - Task allocation under uncertainty
  - Scalability: 100s of robots
- Used by Amazon, Alibaba, JD.com
- MARL methods: MAPPO, QMIX with parameter sharing

Drone Swarm Coordination



- Decentralised control of UAV fleets
- **Applications:**
  - Search & rescue
  - Formation flying
  - Area surveillance
  - Precision agriculture
- Limited communication range (dotted links)
- Each drone: local sensor view (shaded circles)
- MARL learns when/what to communicate
- Methods: CommNet, TarMAC, mean-field MARL

Multi-Agent LLM Systems



Multi-Agent Debate:

- Multiple LLMs argue, critique, refine answers
- Improves reasoning accuracy (Du et al., 2023)
- Connection to MARL: agents optimise for answer quality through interaction

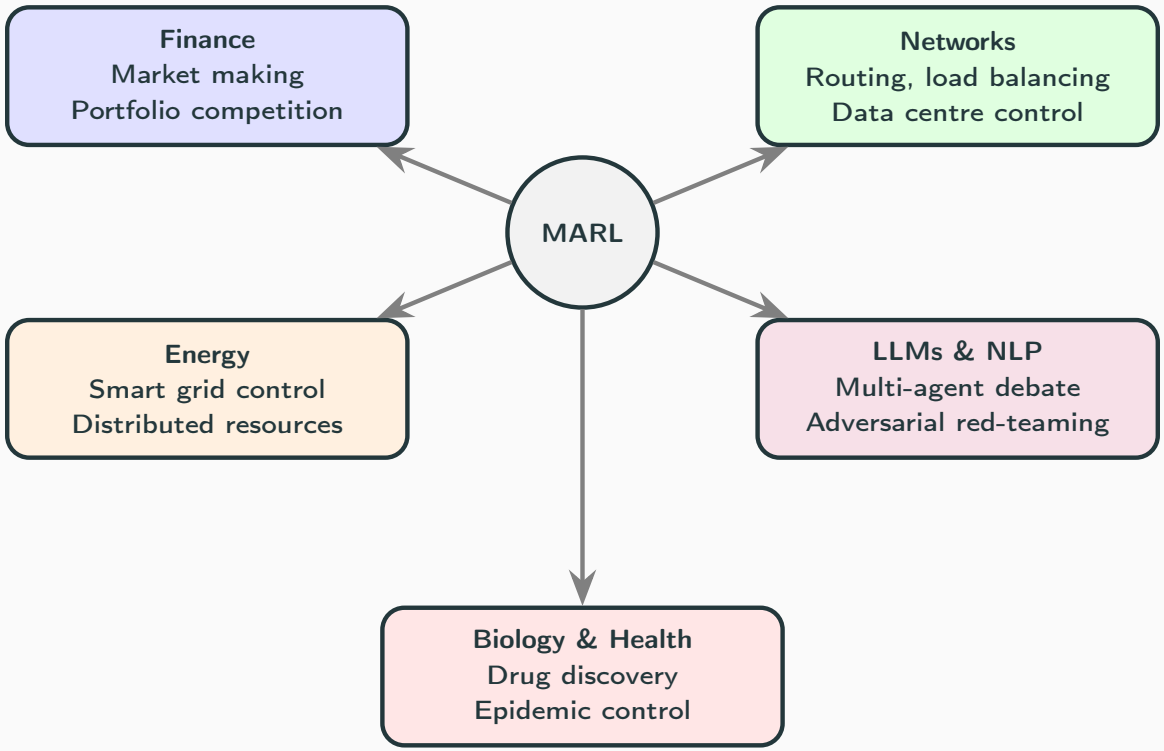
Adversarial Red-Teaming:

- Attacker agent learns to find vulnerabilities
- Defender agent learns to be robust
- Competitive MARL formulation

Agentic Workflows:

- LLM agents with specialised roles (coder, reviewer, tester)
- Cooperative MARL with natural language actions
- Examples: ChatDev, MetaGPT, AutoGen

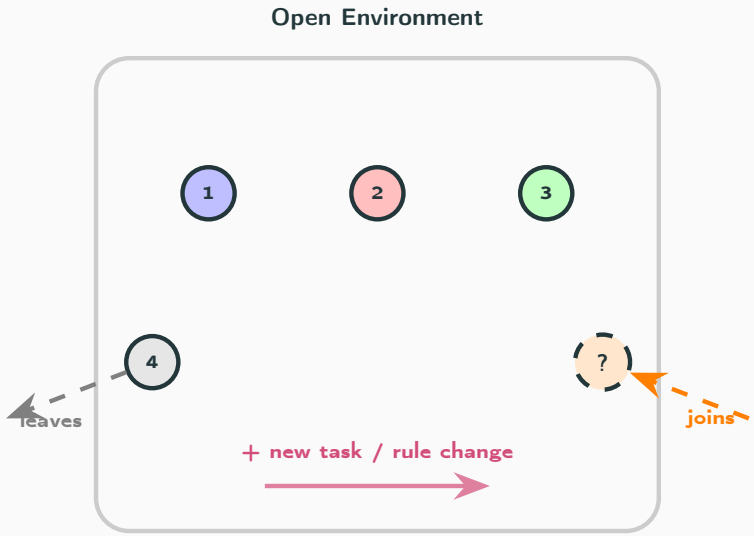
Other Applications



Open Environments in MARL

Closed MARL (standard assumption): fixed set of agents, fixed task, fixed rules.

Open MARL: the environment itself is non-stationary in a structural sense.



Sources of openness:

- **Agent openness**: agents join / leave over time
- **Task openness**: new tasks or goals appear
- **Type openness**: unknown agent types (ad-hoc teammates)
- **Rule openness**: dynamics or reward structure evolve

Key challenges:

- Cannot assume a fixed joint policy space
- Must generalise across *compositions* of agents
- **Zero-shot coordination**: cooperate with unseen partners

Notable work: Melting Pot (DeepMind), Open-Ended Learning (XLand), AdA (Adaptive Agent), Hanabi ad-hoc teamwork challenge.

References

References i

- Shapley, L. (1953). Stochastic games. *PNAS*, 39(10).
- Littman, M. (1994). Markov games as a framework for multi-agent RL. *ICML*.
- Bernstein, D. et al. (2002). The complexity of decentralized control of MDPs. *Mathematics of Operations Research*, 27(4).
- Foerster, J. et al. (2016). Learning to communicate with deep multi-agent RL. *NeurIPS*.
- Sukhbaatar, S. et al. (2016). Learning multiagent communication with backpropagation. *NeurIPS*.
- Lowe, R. et al. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *NeurIPS*.
- Sunehag, P. et al. (2018). Value-decomposition networks for cooperative multi-agent learning. *AAMAS*.
- Rashid, T. et al. (2018). QMIX: Monotonic value function factorisation for deep multi-agent RL. *ICML*.
- Foerster, J. et al. (2018a). Counterfactual multi-agent policy gradients. *AAAI*.
- Foerster, J. et al. (2018b). Learning with opponent-learning awareness (LOLA). *AAMAS*.
- Son, K. et al. (2019). QTRAN: Learning to factorize with transformation for cooperative multi-agent RL. *ICML*.
- Das, A. et al. (2019). TarMAC: Targeted multi-agent communication. *ICML*.
- Vinyals, O. et al. (2019). Grandmaster level in StarCraft II using multi-agent RL. *Nature*.
- Berner, C. et al. (2019). Dota 2 with large scale deep RL. *arXiv:1912.06680*.
- Brown, N. & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science*.

References ii

- Baker, B. et al. (2020). Emergent tool use from multi-agent autocurricula. *ICLR*.
- Rashid, T. et al. (2020). Weighted QMIX: Expanding monotonic value function factorisation. *NeurIPS*.
- Wang, J. et al. (2021). QPLEX: Duplex dueling multi-agent Q-learning. *ICLR*.
- Zhang, K. et al. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of RL and Control*, Springer.
- Yu, C. et al. (2022). The surprising effectiveness of PPO in cooperative multi-agent games. *NeurIPS*.
- Du, Y. et al. (2023). Improving factuality and reasoning in LLMs through multiagent debate. *arXiv:2305.14325*.
- Terry, J. et al. (2021). PettingZoo: Gym for multi-agent RL. *NeurIPS*.

Questions?