Multi-Agent Reinforcement Learning

Weinan Zhang
Shanghai Jiao Tong University
http://wnzhang.net

May 2024

2024年上海交通大学ACM班强化学习课程大纲

强化学习基础部分

（中文课件）

1. 强化学习、探索与利用
2. MDP和动态规划
3. 值函数估计
4. 无模型控制方法
5. 规划与学习
6. 参数化的值函数和策略
7. 深度强化学习价值方法
8. 深度强化学习策略方法

强化学习前沿部分

（英文课件）

9. 基于模型的深度强化学习
10. 模仿学习
11. 离线强化学习
12. 多智能体强化学习基础
13. 多智能体强化学习前沿
14. 基于扩散模型的强化学习
15. AI Agent与决策大模型
16. 技术交流与回顾

Content

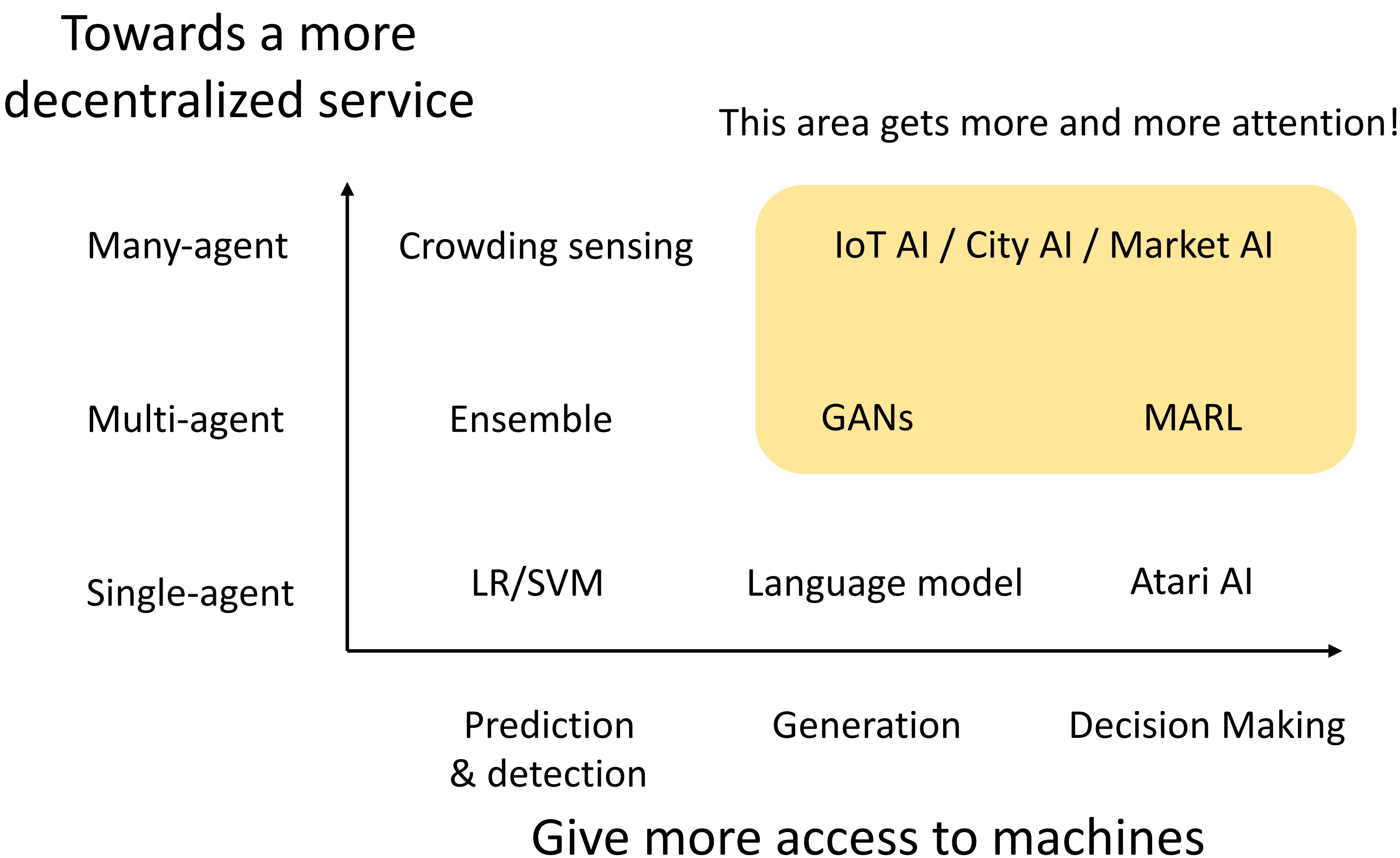
- Background of MARL
- Fundamentals of Game Theory
- Multi-Agent Reinforcement Learning
- Many-Agent Reinforcement Learning

The Coming Intelligent IoT Era

Intelligent Internet of Things based on 5G

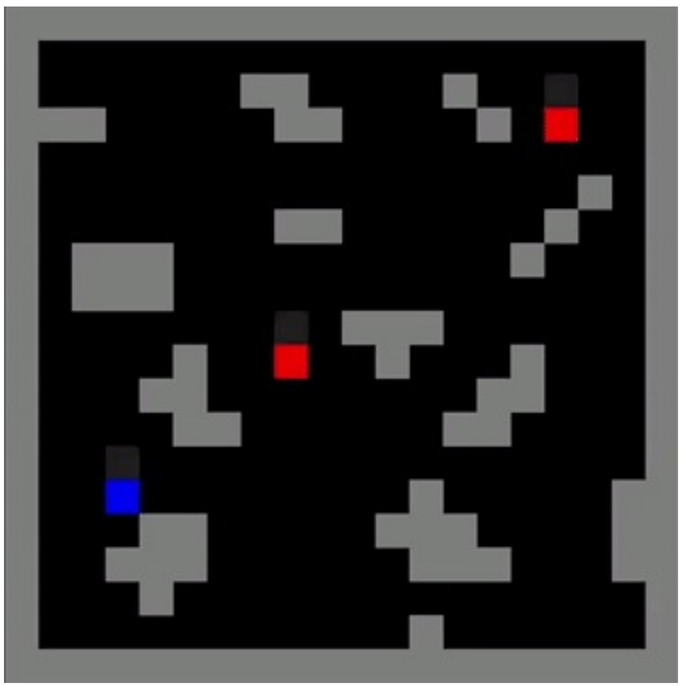
Autonomous driving cars with inter-connections

Observation: Machine Learning Paradigm Extension



MARL Case: Multi-Agent Game Playing

- Multi-agent game playing
  - Learning to cooperate and compete



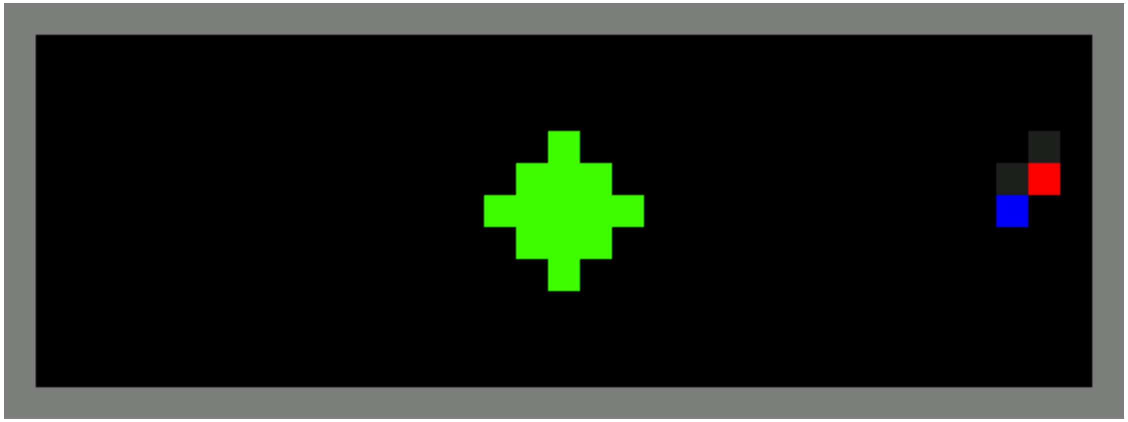
Wolfpack game
- Red agents are the predators
- Blue agent is the prey
- Red agent gets close to blue agent to make a capture, then the whole team gets a reward

Results
- Red agents learn to cooperate.

Leibo, Joel Z., et al. "Multi-agent Reinforcement Learning in Sequential Social Dilemmas." *AAMAS 2017.*

MARL Case: Multi-Agent Game Playing

- Multi-agent game playing
  - Learning to cooperate and compete



Gathering game

- Red and blue agents compete for food
- Each agent can either move to eat or attack the other to make it paused

Results

- Red agents learn to compete (shooting each other) when food resource is insufficient

Leibo, Joel Z., et al. "Multi-agent Reinforcement Learning in Sequential Social Dilemmas." *AAMAS 2017.*

MARL Case: Multi-Agent Game Playing

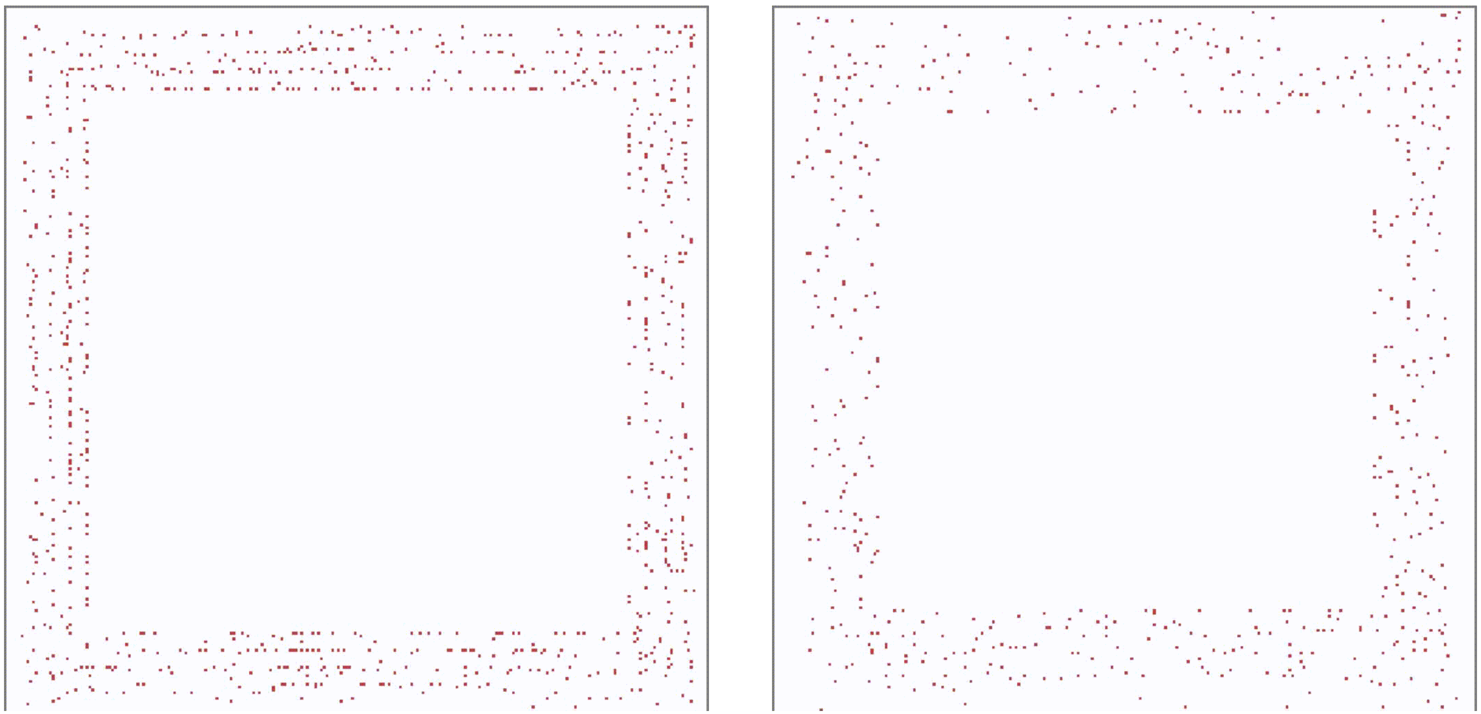
- Multi-agent game playing
  - Learning to cooperate and compete

  - Agents learn to cooperate in a team to fight against another team (here the other team is just hand-crafted AI)

Peng Peng, Jun Wang et al. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. NIPS workshop 2017.

MARL Case: Army Align

- Let an army of agents align a particular pattern

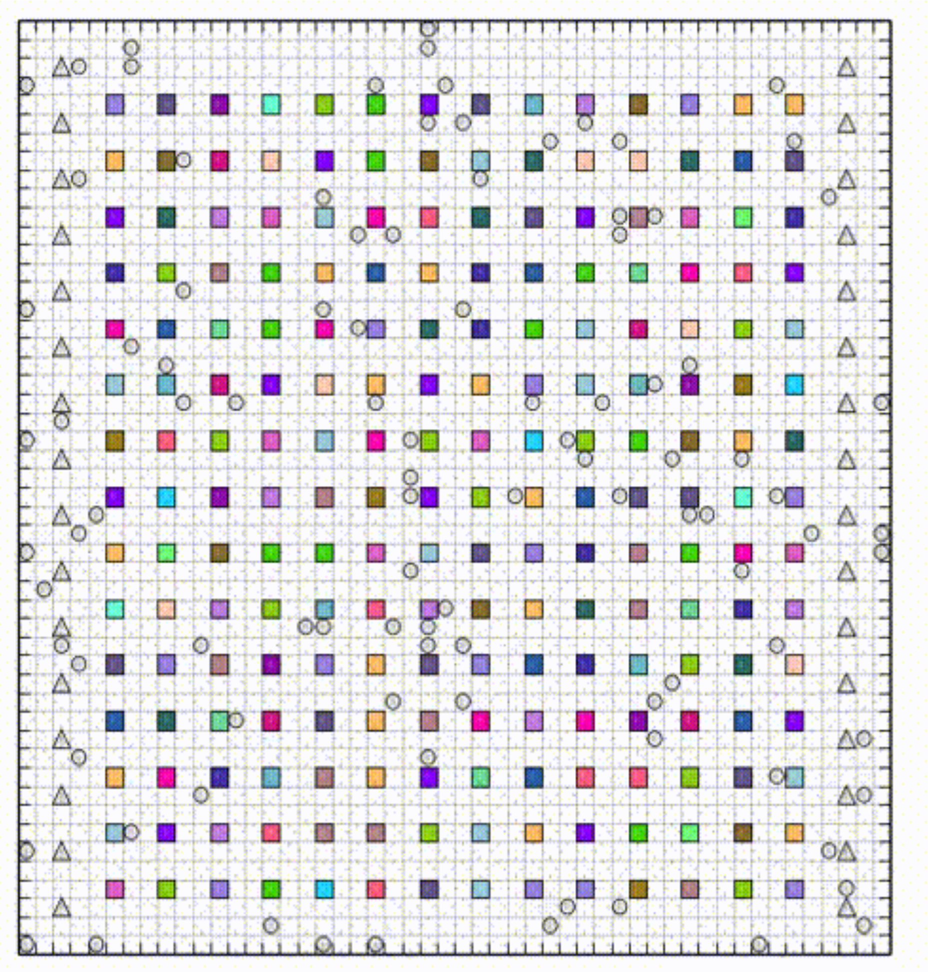


Lianmin Zheng, Weinan Zhang et al. Magent: a many-agent reinforcement learning platform for artificial collective intelligence. NIPS17 & AAAI18.

MARL Case: City Traffic Simulation

- Designing
  - Car routing policy
  - Traffic light controller
  - Fleet management & taxi dispatch
- Shortcomings
  - Discrete implementation is not suitable for traffic simulation

Use Case: Storage Sorting Robots

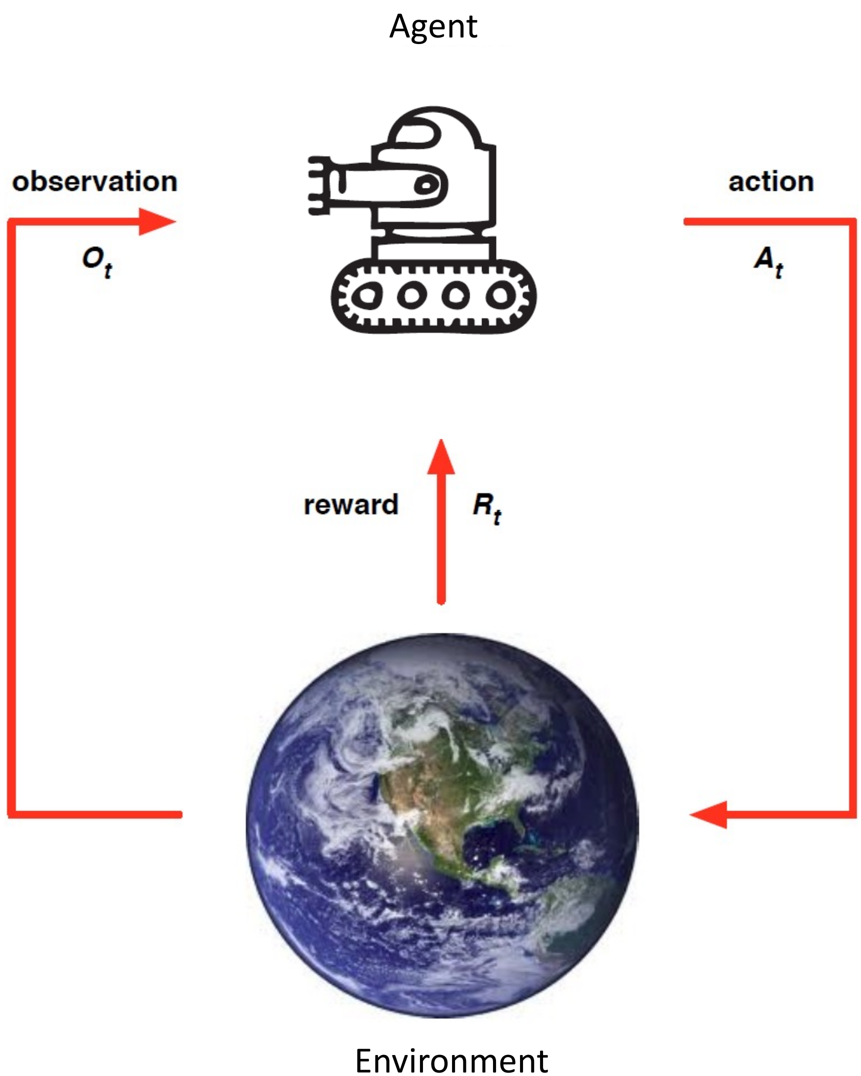


Haifeng Zhang, et al. “Layout Design for Intelligent Warehouse by Evolution with Fitness Approximation.” IEEE Access 2019.

REVIEW

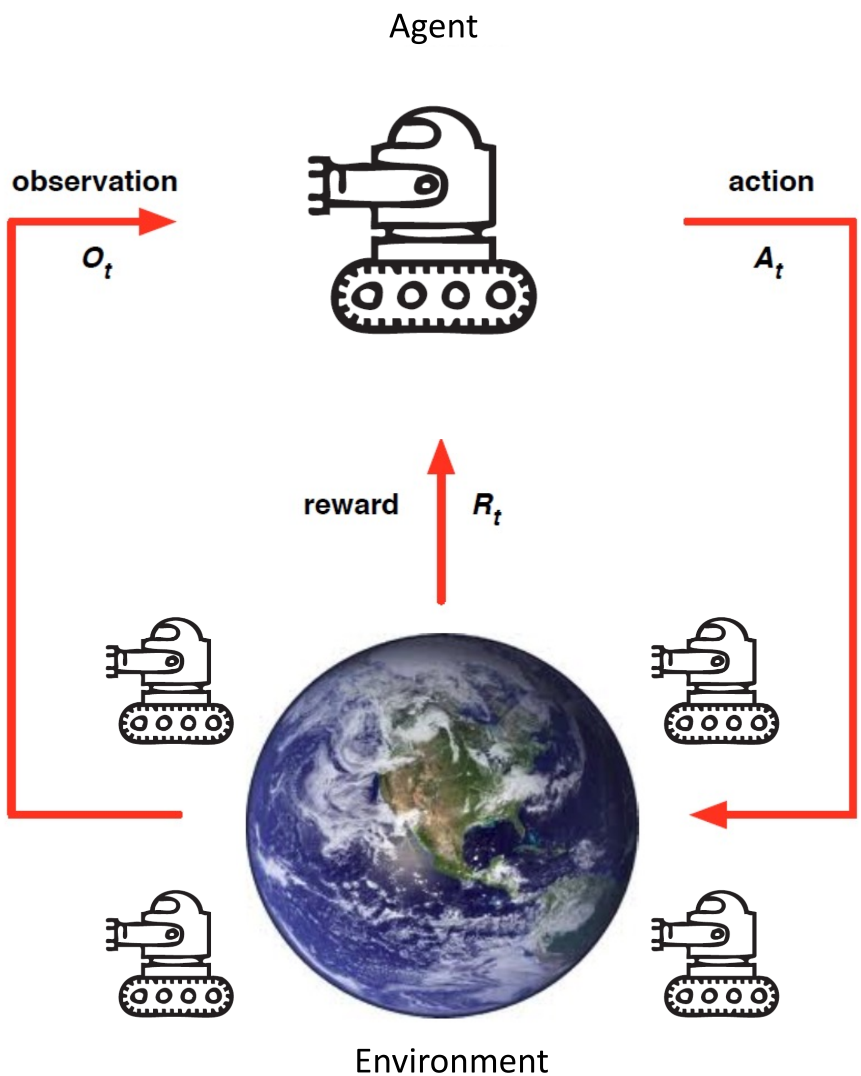
Reinforcement Learning

- Learning from interaction with the environment
- The agent
  - senses the observations from environment
  - takes actions to deliver to the environment
  - gets reward signal from the environment
- Normally, the environment is stationary



Multi-Agent Reinforcement Learning

- Learning from interaction with the environment
- The environment contains other agents that are learning and updating
- Non-stationary environment



Fundamental Difficulty in Multi-Agent RL

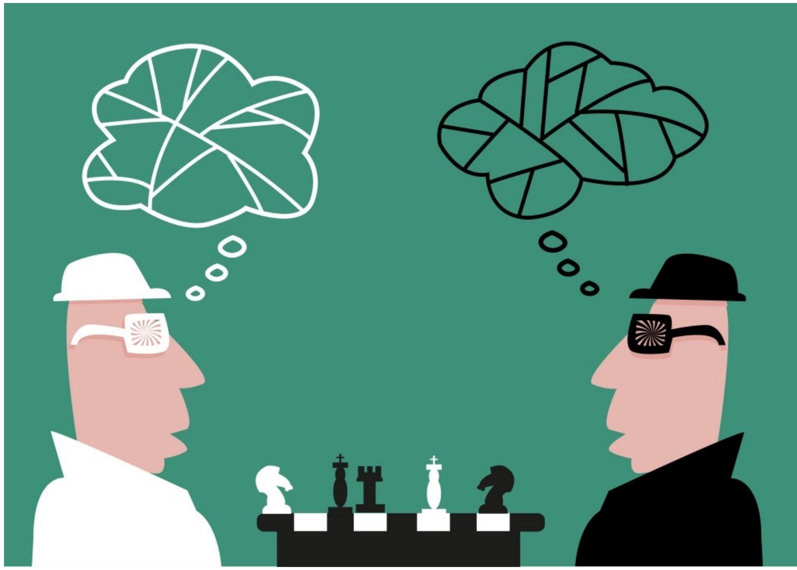
- MARL is fundamentally more difficult since agents not only learn to interact with the environment but also with each other
- Directly applying single-agent RL algorithms will have no guarantee of effectiveness
- Solution: game theory!

Content

- Background of MARL
- Fundamentals of Game Theory
- Multi-Agent Reinforcement Learning
- Many-Agent Reinforcement Learning

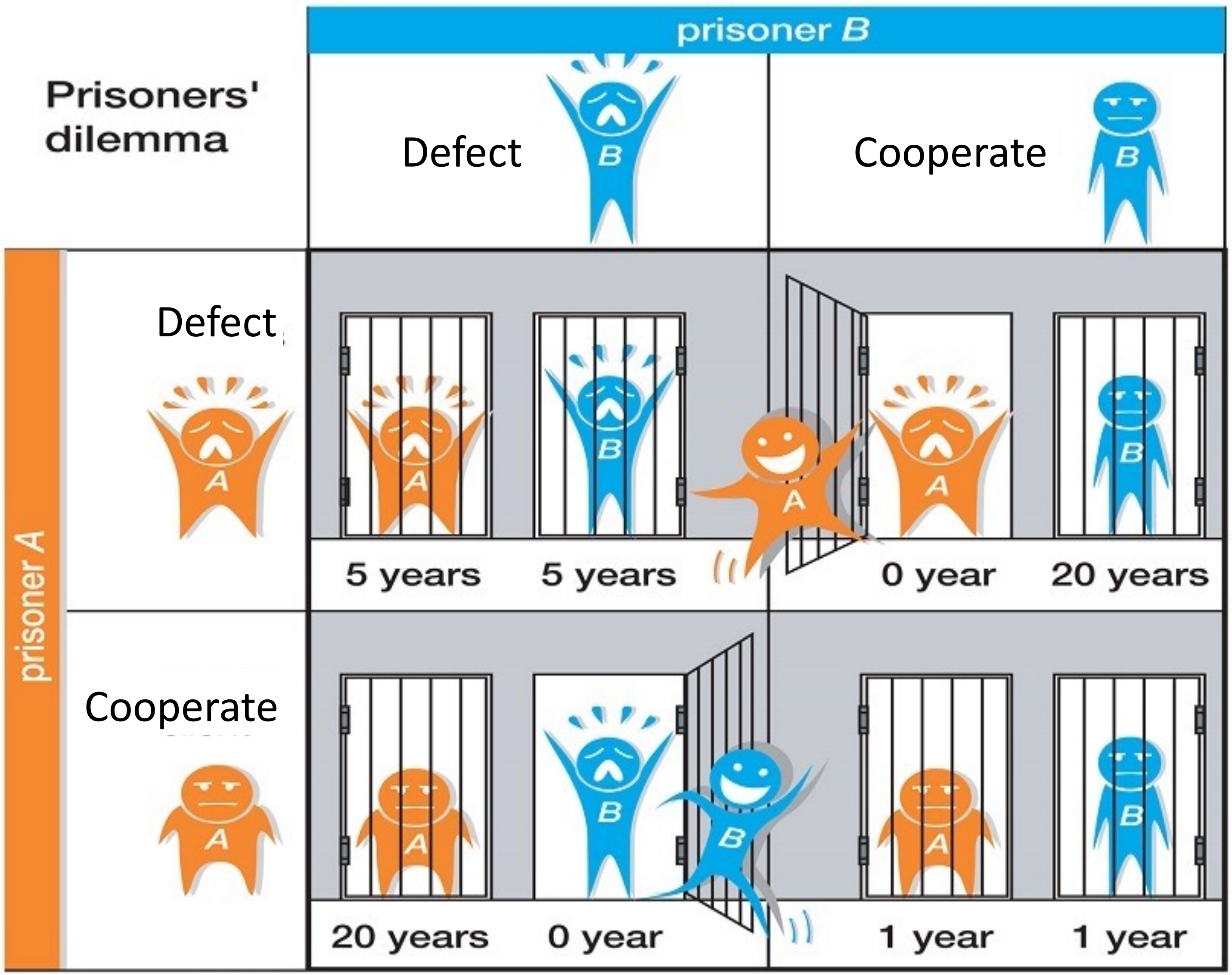
What is Game Theory

- Games theory studies interaction of self-interested agents
  - What does “self-interested” mean?
- Modeling an agent’s interests: utility theory
  - Utility function: mapping from states to real numbers



- Things get more complicated with multiple agents
  - One’s actions can affect others’ utilities
  - Noncooperative game theory – individual

Prisoner's Dilemma



Prisoner's Dilemma

- If two players tune their strategies interactively, they will finally converge to defect-defect action profile

	Defect	Cooperate
Defect	-5, -5	0, -20
Cooperate	-20, 0	-1, -1

	Defect	Cooperate
Defect	a, a	c, b
Cooperate	b, c	d, d

$c > d > a > b$

Normal-form Game (正则形式博弈)

- Also known as the strategic or matrix form.
- It is a representation of every player's utility for every state of the world, where the states of the world depend only on the player's combined actions.

Player 2

Player 1		Action 1	Action 2
	Action 1	a_1, a_2	b_1, b_2
	Action 2	c_1, c_2	d_1, d_2

- Most other representations of games can be reduced to (maybe much larger) normal-form games.

Normal-form Game

Definition A (finite, *n*-player) normal-form game is a tuple (N, A, u), where:

- N is a finite set of *n* players, indexed by i;
- $A = A_1 \times \cdots \times A_n$, where A_i is a finite set of actions available to player i;
- Each vector $a = (a_1, \cdots, a_n) \in A$ is called an action profile;
- $u = (u_1 \times \cdots \times u_n)$ where $u_i: A \to \mathbb{R}$ is a real-valued utility (or payoff) function for player i.

- Standard representation: an n-dimensional matrix.

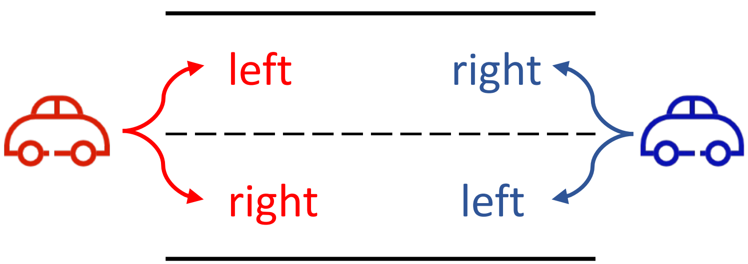
Common-payoff Game

Definition A <u>common-payoff game</u> is a game in which for all action profiles $a \in A_1 \times \cdots \times A_n$ and for any pair of agents i, j, it is the case that $u_i(a) = u_j(a)$.

- Common-payoff games are also called pure coordination games or team games.
- The agents have no conflicting interests.

Coordination Game: Example

- Two drivers driving towards each other in a country having no traffic rules …



	Left	Right
Left	1, 1	0, 0
Right	0, 0	1, 1

Zero-sum Game (零和博弈)

Definition A two-player normal-form game is constant-sum if there exists a constant c such that for each strategy profile $a \in A_1 \times A_2$ it is the case that $u_1(a) + u_2(a) = c$.

- Pure competition
- A constant-sum game is zero-sum if $c = 0$.
- Zero-sum games are most meaningful for two agents because if we allow more agents, any game can be turned into a zero-sum game by adding a dummy player.

Zero-sum Game: Matching Pennies

- Two players present a penny at the same time
  - Player 1 wins if two pennies match
  - Player 2 wins otherwise

	Heads	Tails
Heads	1, -1	-1, 1
Tails	-1, 1	1, -1

Zero-sum Game: Rock, Paper, Scissors

- Two players with three actions

	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

Normal-form Game: Battle of the Sexes

- A couple wishes to go to watch boxing or shopping. They have different preferences but prefer going together.

Wife

Husband

	Boxing	Shopping
Boxing	2, 1	0, 0
Shopping	0, 0	1, 2

- It includes element of both coordination and competition.

Strategies in Normal-Form Games

- Pure strategy: to select a single action and play it.
- Pure-strategy profile: An action profile where each agent plays a pure strategy.

Introducing randomness in the choice of action

- Mixed strategy: randomizing over the set of available actions according to some probability distribution.

Mixed Strategy

Definition Let (N, A, u) be a normal-form game, and for any set X let $\Pi(X)$ be the set of all probability distributions over X. Then the set of mixed strategies for player i is $S_i = \Pi(A_i)$.

Definition The set of mixed-strategy profiles is simply the Cartesian product of the individual mixed-strategy sets, i.e., $S_1 \times \cdots \times S_n$.

Definition The support of a mixed strategy s_i for a player i is the set of pure strategies $\{a_i | s_i(a_i) > 0\}$.

Expected Utility

Definition Given a normal-form game (N, A, u), the expected utility u_i for player i of the mixed-strategy profile $s = (s_1, \ldots, s_n)$ is defined as

$$u_i(s) = \sum_{a \in A} u_i(a) \boxed{\prod_{j=1}^{n} s_j(a_j)} \leftarrow \boldsymbol{\pi}(a)$$

Joint policy

$$a = (a_1, a_2, \ldots, a_n)$$

Best Response

- Formally, define $s_{-i} = (s_i, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$, a strategy profile s without agent i's strategy. Thus we can write $s = (s_i, s_{-i})$.
- If the agents other than i (denoted as $-i$) were to commit to play s_{-i}, what is the best response of agent i?

Definition Player i's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in S_i$ such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$.

Best Response

- The best response is not necessarily unique
  - Some cases – there is a unique best response that is a pure strategy
  - Other cases – the number of best responses is infinite

- If the support of a best response s^* includes more than one actions, the agent must be indifferent among them
  - i.e., the same expected utility $u_i(a_1, s_{-i}) = u_i(a_2, s_{-i})$
  - As such, any blending of a_1 and a_2 is the best response

Nash Equilibrium

Definition

A strategy profile $s = (s_1, \dots, s_n)$ is a Nash equilibrium if, for all agents i, s_i is a best response to s_{-i}.

John Nash

- A Nash equilibrium is a stable strategy profile: no agent would want to change his strategy if he knew what strategies the other agents were following.
- Whether or not every agent's strategy constitutes a unique best response to the other agents' strategies?
  - Yes – strict Nash equilibrium
  - No – weak Nash equilibrium

Finding Nash Equilibrium: Prisoner's Dilemma

- The only Nash equilibrium of prisoner's dilemma is both players defect

	Defect	Cooperate
Defect	-5, -5	0, -20
Cooperate	-20, 0	-1, -1

	Defect	Cooperate
Defect	a, a	c, b
Cooperate	b, c	d, d

c > d > a > b

Finding Nash Equilibria

The Battle of the Sexes game has two pure-strategy Nash equilibria.

Wife

Husband

	Boxing	Shopping
Boxing	2, 1	0, 0
Shopping	0, 0	1, 2

Are these two the only Nash equilibria?

Finding Nash Equilibria

- There is also another mixed-strategy equilibrium.
- Assume that husband's strategy is to watch boxing with probability p and go shopping with probability $1-p$.
- Then if the wife also mixes between boxing and shopping, she must be indifferent between them, given the husband's strategy.

Wife

Husband		Boxing	Shopping
p	Boxing	2, 1	0, 0
$1-p$	Shopping	0, 0	1, 2

$$U_{\text{wife}}(\text{boxing}) = U_{\text{wife}}(\text{shopping})$$
$$1 \times p + 0 \times (1-p) = 0 \times p + 2 \times (1-p)$$
$$\Rightarrow p = \frac{2}{3}$$

Finding Nash Equilibria

The mixed-strategy Nash equilibrium of the Battle of the Sexes game:

- The husband chooses boxing with probability 2/3 and shopping with probability 1/3.
- The wife chooses to boxing with probability 1/3 and shopping with probability 2/3.
- The expected payoff of both players is 2/3 in this equilibrium, so each of the pure-strategy equilibria Pareto-dominates the mixed-strategy equilibrium.

			Wife: Boxing ($\frac{1}{3}$)	Wife: Shopping ($\frac{2}{3}$)
Husband	$\frac{2}{3}$	Boxing	2, 1	0, 0
Husband	$\frac{1}{3}$	Shopping	0, 0	1, 2

Mixed Strategies Matter

What about the Matching Pennies game?

	Heads	Tails
Heads	1, -1	-1, 1
Tails	-1, 1	1, -1

- There is no pure-strategy Nash equilibrium.
- There exists a mixed-strategy equilibrium: each player chooses Heads and Tails with probability 1/2.

Mixed Strategies Matter

For the popular rock-paper-scissors game?

	Rock	Paper	Scissors
Rock	0, 0	-1, 1	1, -1
Paper	1, -1	0, 0	-1, 1
Scissors	-1, 1	1, -1	0, 0

- There is no pure-strategy Nash equilibrium.
- There exists a mixed-strategy equilibrium: each player chooses Rock, Paper and Scissors with probability 1/3.

Existence of Nash Equilibrium

Theorem (Nash, 1951) *Every game with a finite number of players and action profiles has at least one Nash equilibrium.*

- The proof of this theorem is achieved by appealing to *fixed-point theorem*.
- This theorem depends critically on the availability of mixed strategies to the agents.

Thinking on Nash Equilibrium and MARL

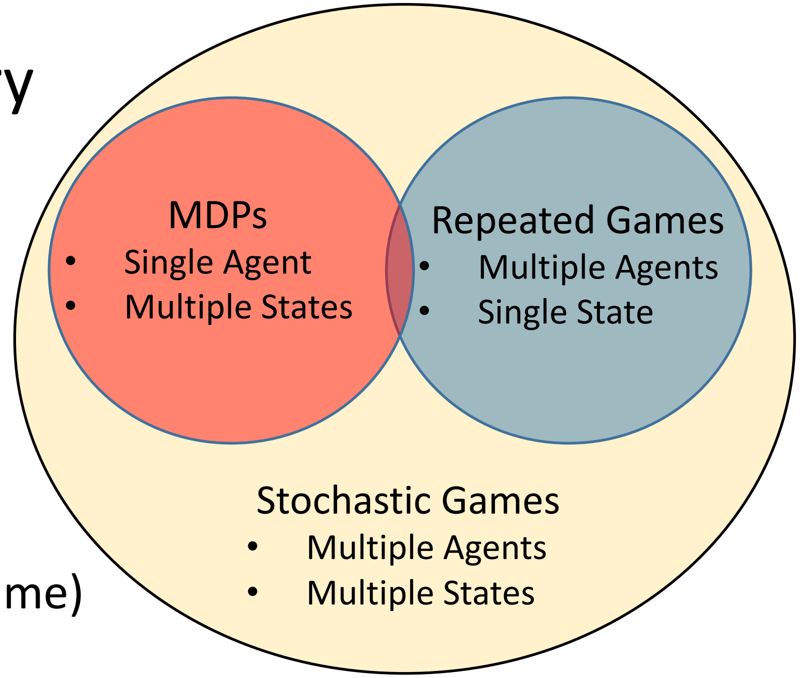
- Given a normal-form game, every player *i*'s utility depends on the joint strategy profile (s_i, s_{-i}), which makes the multi-agent decision making unstable
- Nash equilibrium provides a peaceful place in such an 'unstable' environment, where no player would want to further change the strategy once getting to the equilibrium
- Nash equilibrium of the normal-form game can be set as the learning target of MARL
- Now we need to consider the case of multiple states

Content

- Introduction to Reinforcement Learning
- Fundamentals of Game Theory
- Multi-Agent Reinforcement Learning
- Many-Agent Reinforcement Learning

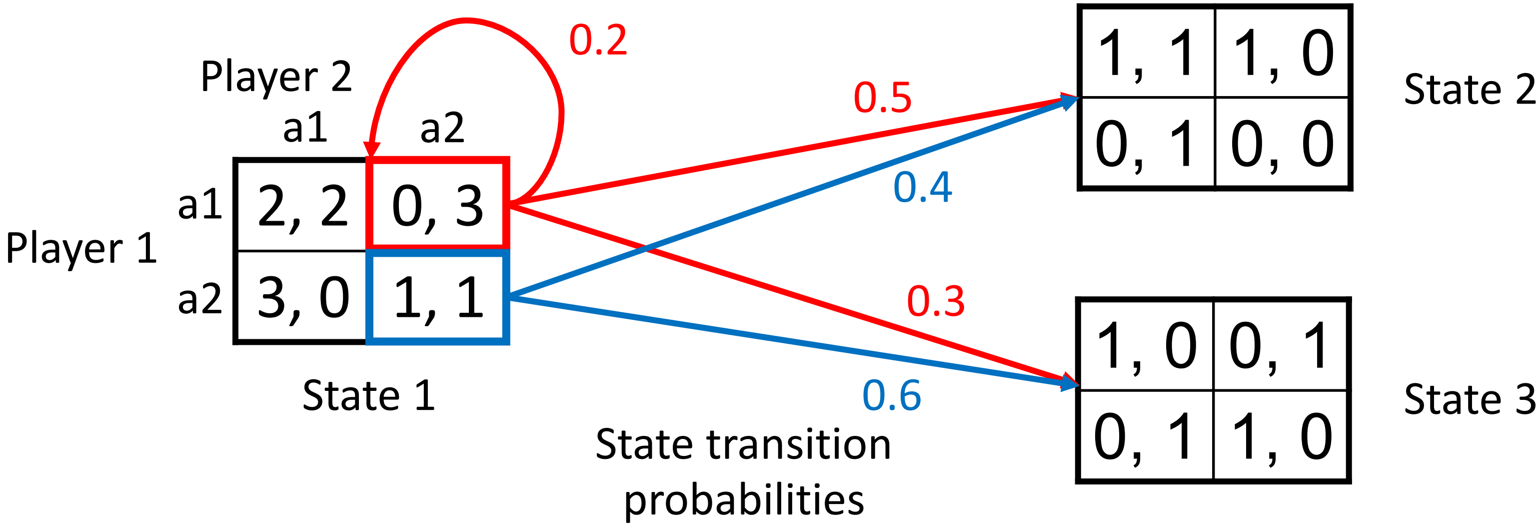
Sequential Decision Making

- 3 types of setting in Game Theory
  - Markov decision processes
    - one decision maker
    - multiple states
  - Repeated games
    - multiple decision makers
    - one state (e.g., one normal form game)
  - Stochastic games (Markov games)
    - multiple decision makers
    - multiple states (e.g., multiple normal form games)



Stochastic Games

- A stochastic game has multiple states and multiple agents
  - Each state corresponds to a normal-form game
  - After a round, the game randomly transits to another state
  - Transition probabilities depend on state and joint actions taken by all agents
- Typically, rewards are discounted over time



Shapley, Lloyd S. "Stochastic games." *Proceedings of the national academy of sciences* 39.10 (1953): 1095-1100.

Definition of Stochastic Games

- A stochastic game is defined by

$$(\mathcal{S}, \mathcal{A}^1, \ldots, \mathcal{A}^N, r^1, \ldots, r^N, p, \gamma)$$

- State space: $\mathcal{S}$
- Action space of agent *j*: $\mathcal{A}^j, \;\; j \in \{1, \ldots, N\}$
- Reward function of agent $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$
- Transition probability $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Omega(\mathcal{S})$

The collection of probability distributions over *S*

- Discount factor across time $\gamma \in [0, 1)$

Policies in Stochastic Games

- For agent *j*, the corresponding policy is

$$\pi^j : \mathcal{S} \rightarrow \Omega(\mathcal{A}^j)$$

The collection of probability distributions over A^j

- The joint policy of all agents is $\boldsymbol{\pi} \triangleq [\pi^1, \ldots, \pi^N]$

- State value function of agent *j*

$$v^j_{\boldsymbol{\pi}}(s) = v^j(s; \boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\boldsymbol{\pi},p}\left[r^j_t | s_0 = s, \boldsymbol{\pi}\right].$$

- Action value function of agent *j* $Q^j_{\boldsymbol{\pi}} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$

$$Q^j_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r^j(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s' \sim p}[v^j_{\boldsymbol{\pi}}(s')]$$

$$[a^1, \ldots, a^N]$$

Independent Learning in SG

- For each agent *j*, assume the other agents' policies are stationary, thus the environment for *j* is stationary to perform Q-learning

$$Q^j(s,a^j) \leftarrow Q^j(s,a^j) +$$
$$\alpha(r^j(s,a^j,a^{-j}) + \gamma \max_{a^{j\prime}} Q^j(s',a^{j\prime}) - Q^j(s,a^j))$$

  - The agent does not know opponents' actions, thus may use the last-step actions or build opponent models
  - Unfortunately, in SG with MARL, every agent is learning and updating its policy, making the environment non-stationary

Nash Equilibrium in SG

$$v^j_{\boldsymbol{\pi}}(s) = v^j(s;\boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\boldsymbol{\pi},p}\left[r^j_t | s_0 = s, \boldsymbol{\pi}\right]$$

- Optimizing $v^j_{\boldsymbol{\pi}}(s)$ for agent *j* depends on the joint policy $\boldsymbol{\pi}$

- Nash equilibrium in SG is represented by a particular joint policy

$$\boldsymbol{\pi}_* \triangleq [\pi^1_*, \ldots, \pi^N_*]$$

  such that nobody would like to change his policy given the others'

$$v^j(s;\boldsymbol{\pi}_*) = v^j(s;\pi^j_*, \boldsymbol{\pi}^{-j}_*) \geq v^j(s;\pi^j, \boldsymbol{\pi}^{-j}_*)$$

$$\boldsymbol{\pi}^{-j}_* \triangleq [\pi^1_*, \ldots, \pi^{j-1}_*, \pi^{j+1}_*, \ldots, \pi^N_*]$$

Nash Q-learning

- Given a Nash policy $\boldsymbol{\pi}_*$, the Nash value function

$$\boldsymbol{v}^{\mathbf{Nash}}(s) \triangleq [v^1_{\boldsymbol{\pi}_*}(s), \ldots, v^N_{\boldsymbol{\pi}_*}(s)]$$

- Nash Q-learning defines an iterative procedure
  1. Solving the Nash equilibrium $\boldsymbol{\pi}_*$ of the current stage defined by $\{\boldsymbol{Q}_t\}$
  2. Improving the estimation of the Q-function with the new Nash value $\boldsymbol{v}^{\text{Nash}}$

- But Nash Q-learning suffers from
  - Very high computational complexity
  - May not work when other agents' policy is unavailable

Hu, Junling, and Michael P. Wellman. "Nash Q-learning for general-sum stochastic games." *Journal of machine learning research* 4.Nov (2003): 1039-1069.

Nash Q-learning

Initialize $Q(s, a)$ arbitrarily

Initialize s

loop

$a_i \leftarrow$ probabilistic outcome of Nash policy derived from $Q(s, a)$, for player i {Mixed with exploration policy}

Take action a_i, observe reward r, next state s' and the joint action of other players a_{-i}

for $i = 1 \ldots n$ **do**

$$Q_i(s, \langle a_i, a_{-i}\rangle) \leftarrow Q_i(s, \langle a_i, a_{-i}\rangle) + \alpha\big(r_i + \boxed{\gamma V_i(s')} - Q_i(s, \langle a_i, a_{-i}\rangle)\big)$$

end for

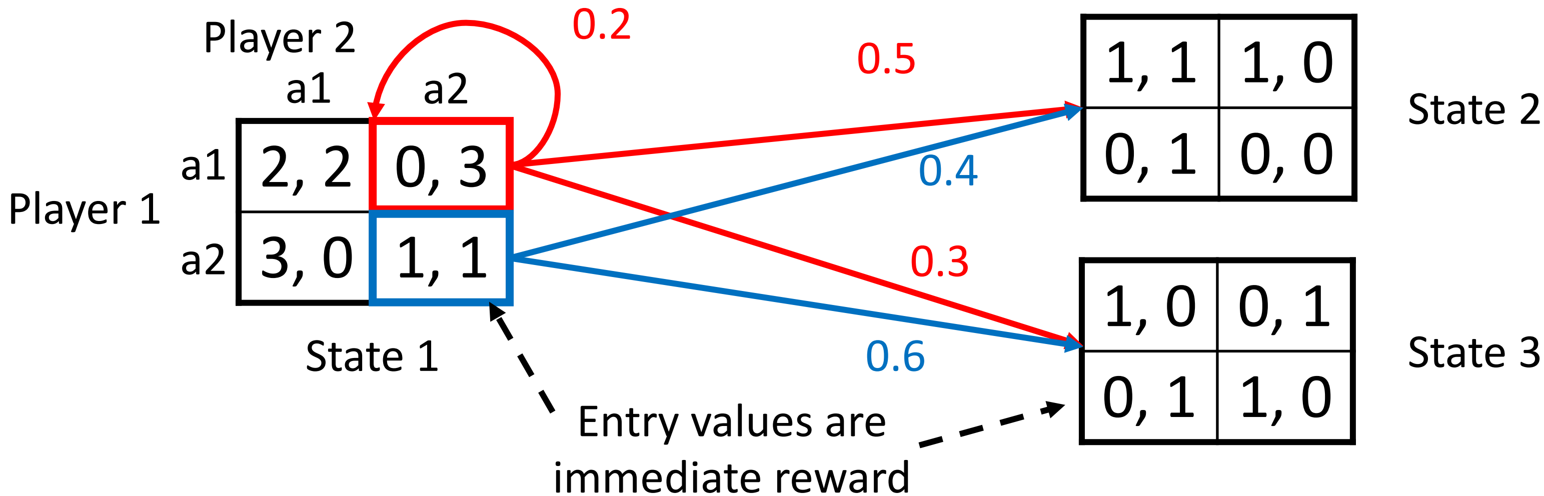
where $\boxed{V(s) = Nash\left([Q(s, a)]\right)}$ Instead of taking "max" as in Q learning

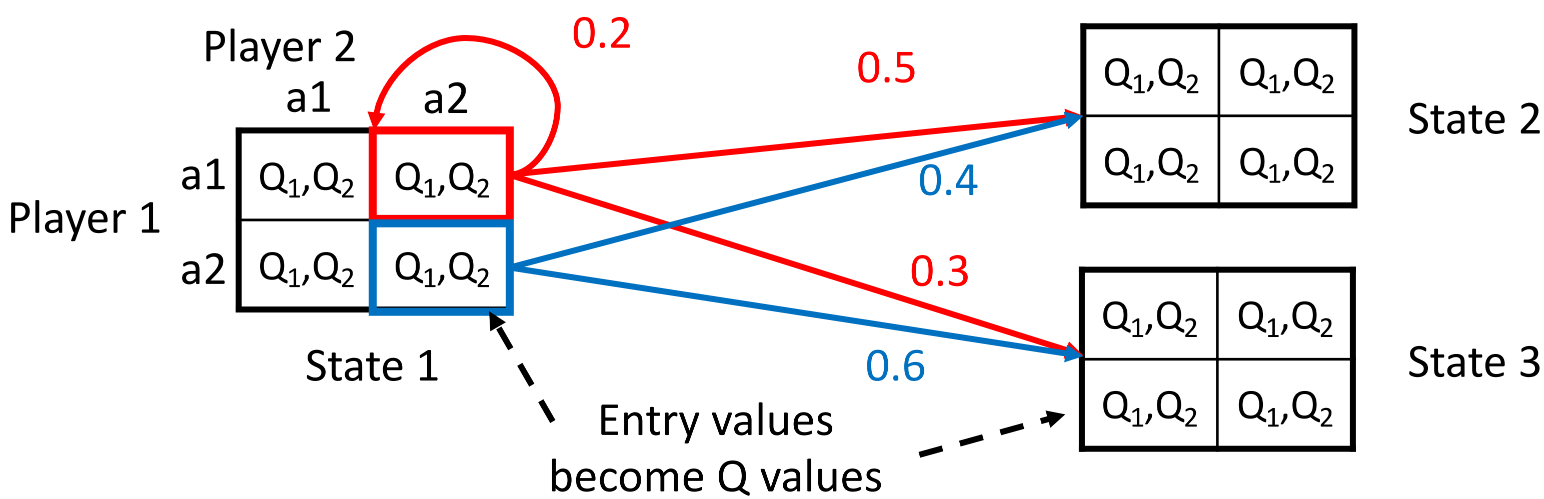
$s \leftarrow s'$

end loop

Hu, Junling, and Michael P. Wellman. "Nash Q-learning for general-sum stochastic games." *Journal of machine learning research* 4.Nov (2003): 1039-1069.

Nash Q-learning





$$Q_i(s, \langle a_i, a_{-i} \rangle) \leftarrow Q_i(s, \langle a_i, a_{-i} \rangle) + \alpha\big(r_i + \gamma V_i(s') - Q_i(s, \langle a_i, a_{-i} \rangle)\big)$$

Minimax Q-learning

- Minimax-Q is designed to work with zero-sum stochastic games
  - in zero-sum games a Nash equilibrium can be found using linear programming

$$V(s) \triangleq \max_{\pi \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a') Q(s, \langle a', o' \rangle)$$

Probability distribution of actions

Own action

Opponent action(s)

Littman, Michael L. "Markov games as a framework for multi-agent reinforcement learning." *Machine Learning Proceedings 1994*. 1994. 157-163.

Minimax Q-learning

Initialize $Q(s, \langle a, o \rangle)$ and $\pi(s)$ arbitrarily

Initialize s
loop
$a \leftarrow$ probabilistic outcome of $\pi(s)$ {Mixed with exploration policy}

Take action a, observe reward r, next state s' and opponent action o

$$Q(s, \langle a, o \rangle) \leftarrow Q(s, \langle a, o \rangle) + \alpha\big(r + \gamma V(s') - Q(s, \langle a, o \rangle)\big)$$
with $\boxed{V(s) = \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a')\, Q(s, \langle a', o' \rangle)}$
$$\pi(s) \rightarrow \arg \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a')\, Q(s, \langle a', o' \rangle)$$

$s \leftarrow s'$
end loop

- *a*: own actions,
- *o*: opponent actions
- *PD*(*A*): prob. distribution of actions

Littman, Michael L. "Markov games as a framework for multi-agent reinforcement learning." *Machine Learning Proceedings 1994*. 1994. 157-163.

Recent Progress of MARL

- Communications between agents
  - Build local communication schemes between agents via the hidden states of deep neural networks
  - CommNet, BiCNet etc.
- Centralized training & decentralized execution
  - Train a centralized critic to guide the update of each actor policy, and execute the actor policies in a decentralized way
  - COMA, MADDPG, QMIX etc.
- Opponent modeling
  - Observe and predict the actions of other agents, so as to perform better decision making
  - LOLA, PR2, ROMMEO etc.
- And many other aspects such as bi-level opt., signal coordination, “win or learning fast” (WoLF) etc.

Training Paradigms of MARL

Fully Decentralized

- Each agent independently senses the local env. and learns its policy
- Like multiple single-RL tasks
- E.g., Independent Q learning

Fully Centralized

- Training and execution are both centralized
- All agents sync at each step, which is costly
- E.g., Single Q learning, CommNet

Centralized Training & Decentralized Execution

- Train the agents together but execute each of them independently
- Agents number and indices are fixed
- E.g., COMA, MADDPG, QMIX

Decentralized Training with Networked Agents

- Agents senses the local env. but can locally sync their info over the network neighbors
- Robust over time-varying network
- E.g., AC with networked agents

Decentralized Training & Execution

Independent Q-Learning

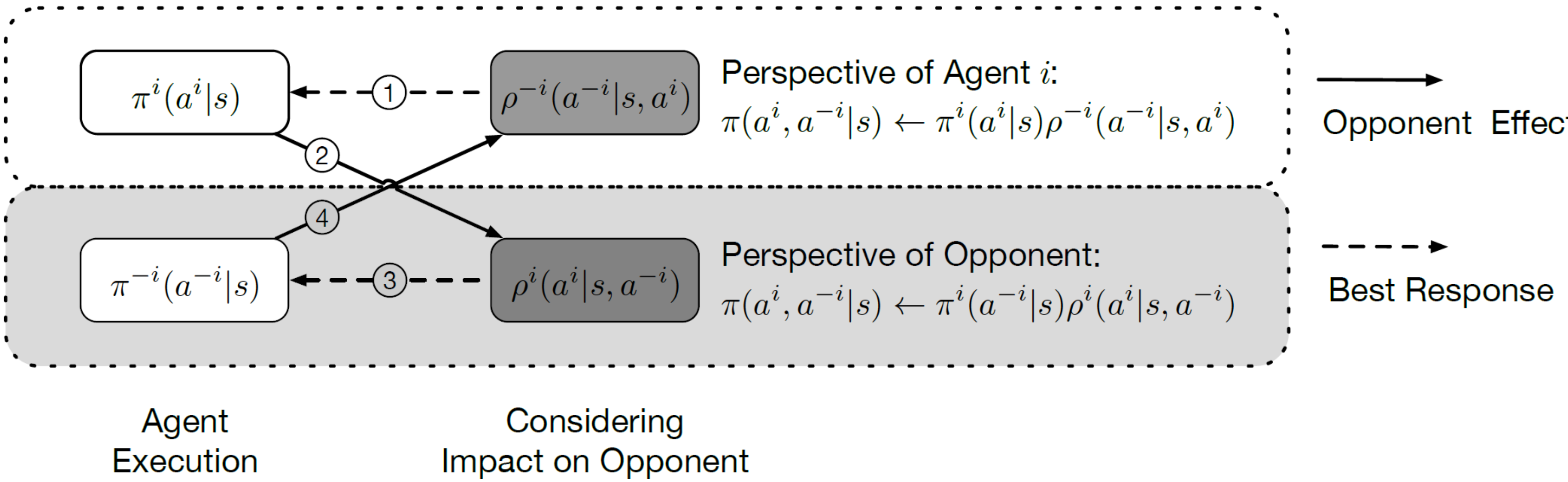
- For each agent *j*, assume the other agents' policies are stationary, thus the environment for *j* is stationary to perform Q-learning

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

- Unfortunately, in SG with MARL, every agent is learning and updating its policy, making the environment non-stationary

Tan, Ming. "Multi-agent reinforcement learning: Independent vs. cooperative agents." *ICML*. 1993.

PR2: Probabilistic Recursive Reasoning for MARL



Recursive reasoning

$$\pi_\theta(a^i, a^{-i}|s) = \underbrace{\pi^i_{\theta^i}(a^i|s)\pi^{-i}_{\theta^{-i}}(a^{-i}|s, a^i)}_{\text{Agent } i\text{'s perspective}} = \underbrace{\pi^{-i}_{\theta^{-i}}(a^{-i}|s)\pi^i_{\theta^i}(a^i|s, a^{-i})}_{\text{The opponents' perspective}}$$

Policy objective

$$\underset{\theta^i, \phi^{-i}}{\arg\max}\ \eta^i\left(\pi^i_{\theta^i}(a^i|s)\rho^{-i}_{\phi^{-i}}(a^{-i}|s, a^i)\right)$$

Policy gradient

$$\nabla_{\theta^i}\eta^i = \mathbb{E}_{s\sim p, a^i\sim\pi^i}\left[\nabla_{\theta^i}\log\pi^i_{\theta^i}(a^i|s)\int_{a^{-i}}\pi^{-i}_{\theta^{-i}}(a^{-i}|s, a^i)Q^i(s, a^i, a^{-i})\,\mathrm{d}a^{-i}\right]$$

Wen, Ying, et al. "Probabilistic recursive reasoning for multi-agent reinforcement learning." *ICLR 2019.*

Decentralized Training & Execution (with Opponent Modeling)

LOLA: Learning with Opponent-Learning Awareness

- Main idea: not only consider the opponent's current policy, but further consider how it will change for the next step!
- Naïve learner:

$$\theta^1_{i+1} = \text{argmax}_{\theta^1} V^1(\theta^1, \theta^2_i)$$
$$\theta^2_{i+1} = \text{argmax}_{\theta^2} V^2(\theta^1_i, \theta^2)$$
$$\theta^1_{i+1} = \theta^1_i + \nabla_{\theta^1_i} V^1(\theta^1_i, \theta^2_i) \cdot \delta$$

- LOLA learner optimizes

$$\max_{\theta^1} V^1(\theta^1, \theta^2 + \Delta\theta^2) \approx V^1(\theta^1, \theta^2) + (\Delta\theta^2)^T \nabla_{\theta^2} V^1(\theta^1, \theta^2)$$

Consider opponent policy learning

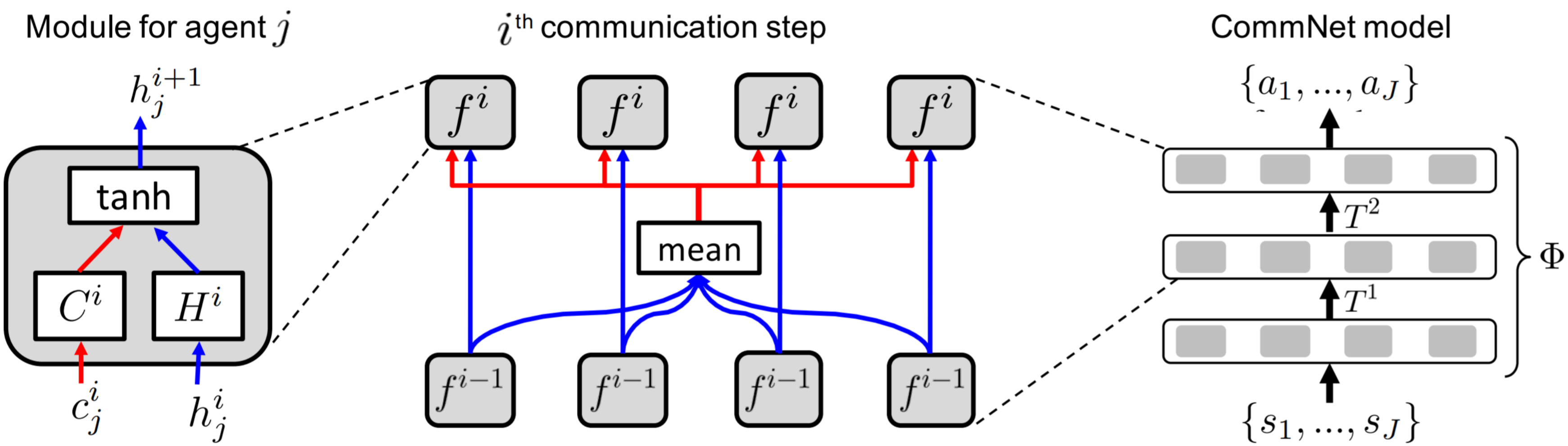
First-order Taylor expansion approximation

Foerster, Jakob, et al. "Learning with opponent-learning awareness." *AAMAS* 2018.

Fully Centralized

CommNet: Learnable Communication Scheme

- Design learnable explicit communication scheme is important for directly achieving agent coordination



$$h_j^{i+1} = f^i(h_j^i, c_j^i)$$

$$c_j^{i+1} = \frac{1}{|N(j)|} \sum_{j' \in N(j)} h_{j'}^{i+1}$$

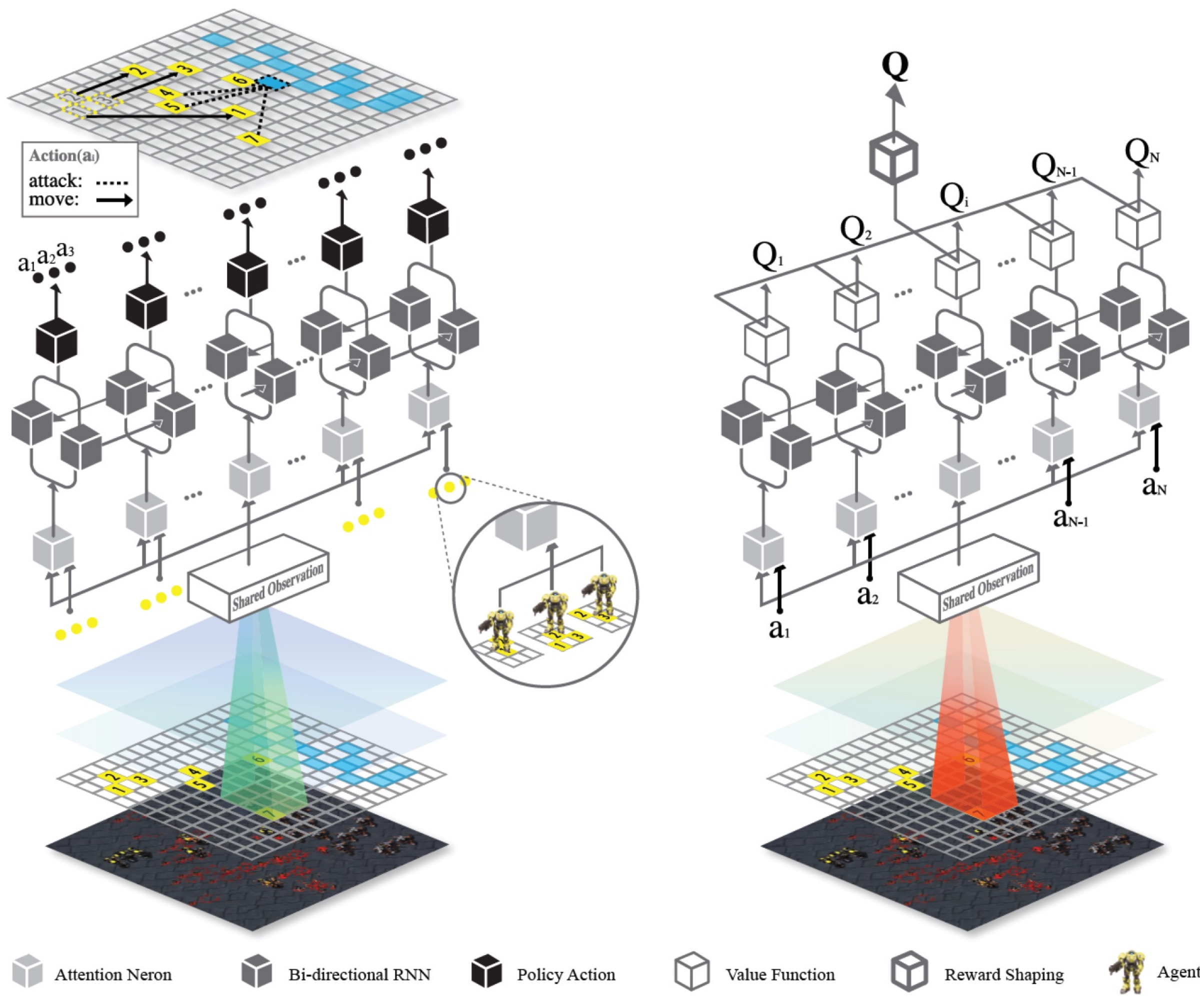
Possible skip connections:

$$h_j^{i+1} = f^i(h_j^i, c_j^i, h_j^0)$$

Sukhbaatar et al. Learning Multiagent Communication with Backpropagation. NIPS 2016.

Fully Centralized

BiCNet: Bidirectionally-Coordinated Net for MARL



(a) Multiagent policy networks with grouping (b) Multiagent Q networks with reward shaping

Peng Peng, Jun Wang et al. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. NIPS workshop 2017.

Fully Centralized

BiCNet: Bidirectionally-Coordinated Net for MARL

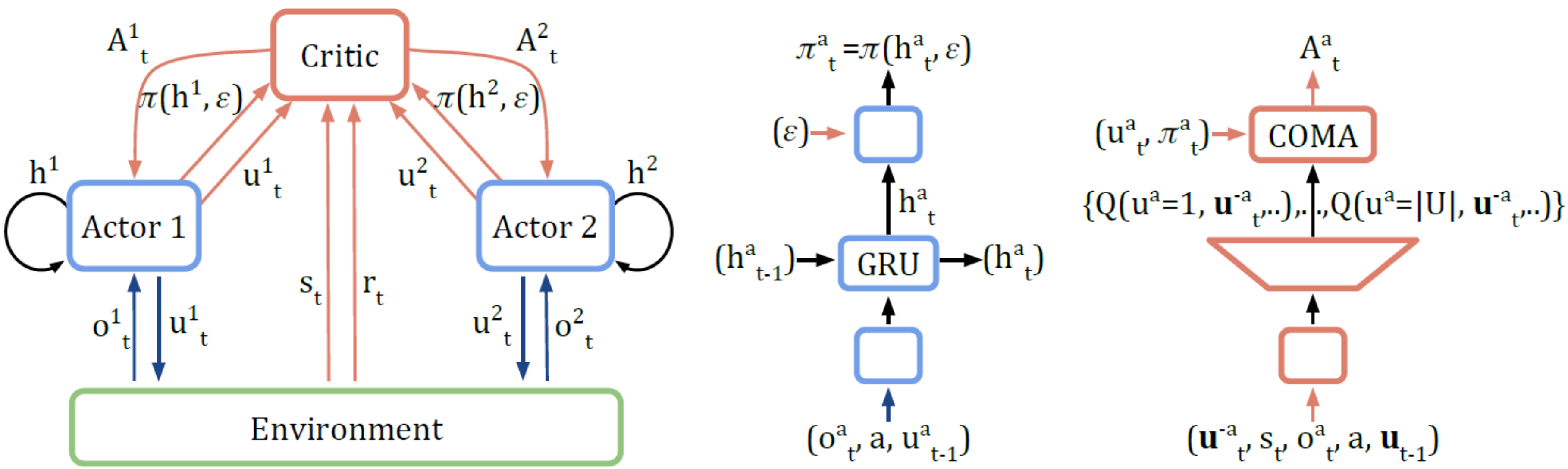
- Multi-agent game playing
  - Learning to cooperate and compete

  - Agents learn to cooperate in a team to fight against another team (here the other team is just hand-crafted AI)

Peng Peng, Jun Wang et al. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. NIPS workshop 2017.

Centralized Training & Decentralized Execution

COMA: Counterfactual Multi-Agent PG



COMA Model Actor Critic

- **Actor-critic policy gradient**

$$J(\theta) = \mathbb{E}_{\pi_\theta}[R_0] \qquad \nabla_\theta J(\theta) = \mathbb{E}_{s_{0:\infty}, u_{0:\infty}}\left[\sum_{t=0}^{T} A^a(s, \mathbf{u}) \nabla_\theta \log \pi(u_t | s_t)\right]$$

- **Counterfactual advantage function**

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

Foerster, Jakob N., et al. "Counterfactual multi-agent policy gradients." *AAAI* 2018.

Centralized Training & Decentralized Execution

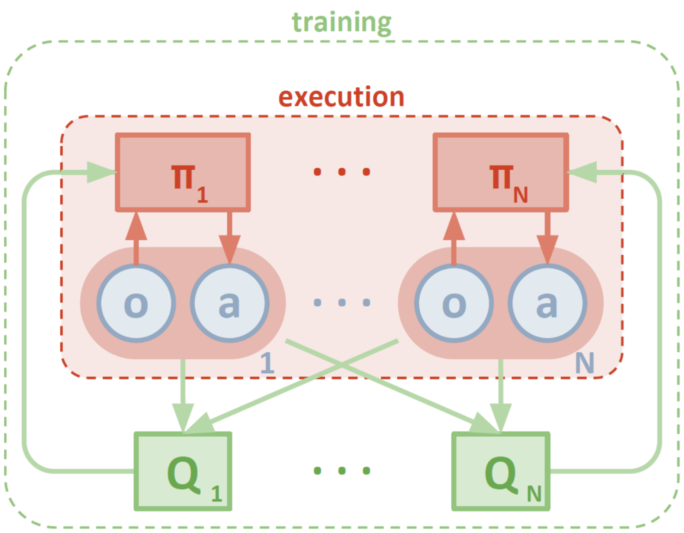
MADDPG: Multi-Agent DDPG

- Centralized action-value function

$$Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, ..., a_N)$$

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2]$$

$$y = r_i + \gamma\, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}', a_1', \ldots, a_N')\big|_{a_j'=\boldsymbol{\mu}_j'(o_j)}$$



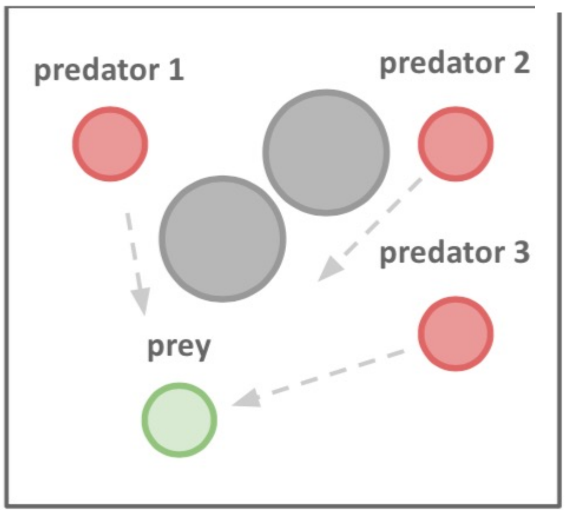
- Deterministic policy gradient via chain rule

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}[\nabla_{\theta_i}\boldsymbol{\mu}_i(a_i|o_i)\nabla_{a_i}Q_i^{\boldsymbol{\mu}}(\mathbf{x}, a_1, ..., a_N)|_{a_i=\boldsymbol{\mu}_i(o_i)}]$$

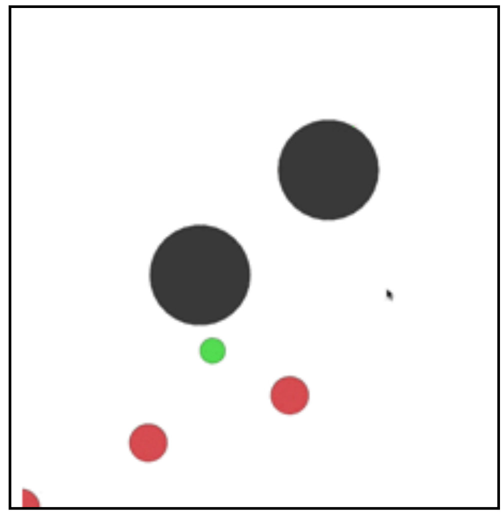
- Comparing with COMA, MADDPG
  - Learn a centralized critic for the agents
  - Continuous policies

Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *NIPS* 2017.

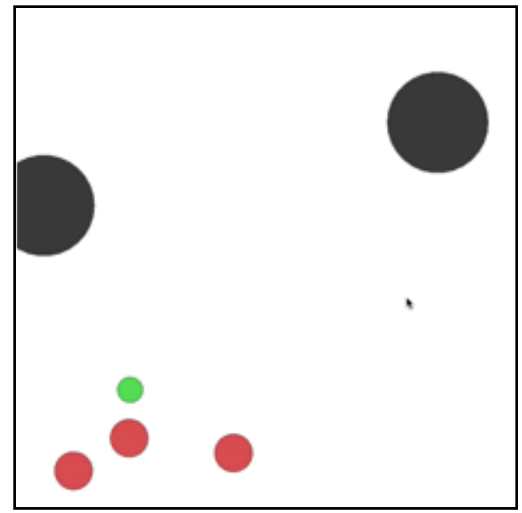
Learning Multi-Agent Interactions



Predator-prey task

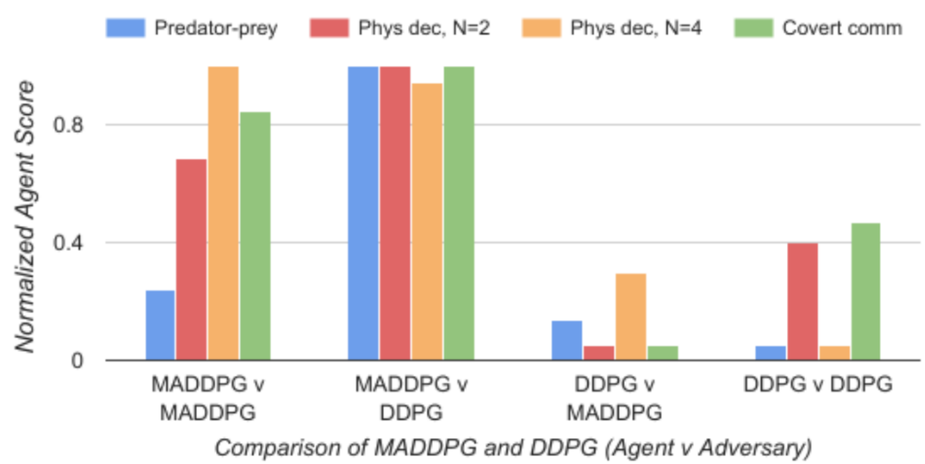


DDPG



MADDPG

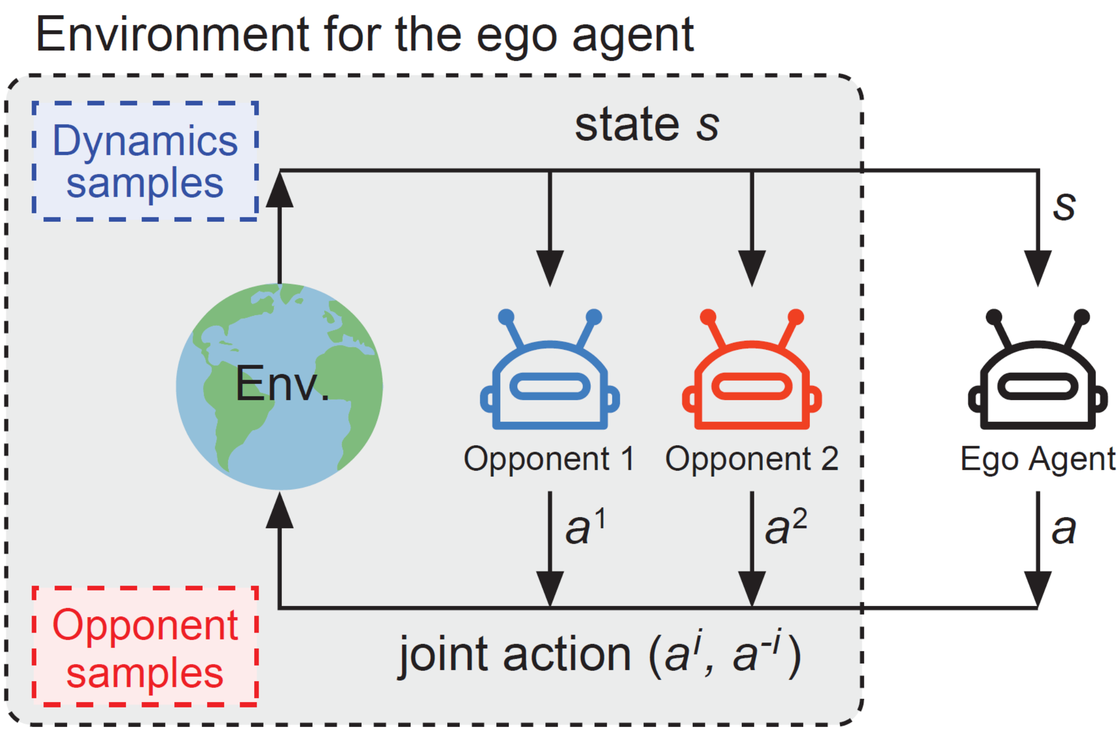
- Evaluation protocol
  - A vs. B for training & test
  - If A gets higher score in A-vs-B than A-vs-A, then it means A is better than B



Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." *NIPS* 2017.

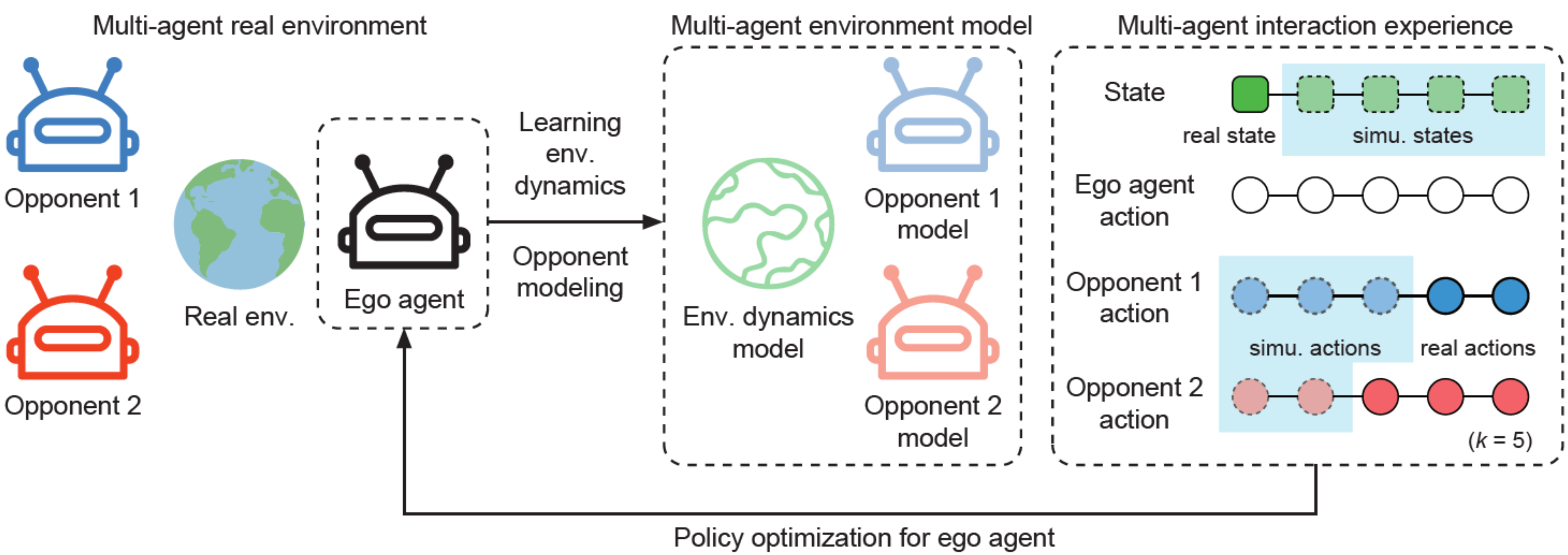
Decentralized Training & Execution (with Models)

Two Parts of Sample Complexity in MARL



- Dynamics sample complexity: num. of real environment transitions (observations sampled from $p(s'|s, a_1, \ldots, a_n)$)
- Opponent sample complexity: num. of real opponent actions (observations sampled from $\pi_j(a|s)$)

Decentralized Model-based MARL



- Multi-agent environment model
  - Environment dynamics model $\mathcal{T}(s'|s, a^i, a^{-i})$
  - Opponent model $\pi^{-i}(a^{-i}|s)$
- Decentralized MARL: each agent independently maintains its multi-agent environment model as above

$-i = \{j\}_{j \neq i}$

Weinan Zhang et al. Model-based Multi-agent Policy Optimization with Adaptive Opponent-wise Rollouts. IJCAI 2021.

Bound of Policy Value Discrepancy

- Multi-agent branched rollout scheme

1. Learn environment dynamics $\hat{\mathcal{T}}$ and opponent models $\hat{\pi}^{-i}$
2. Start from an experienced state and start the rollout with ego policy π^i and above models to collect simulated data
3. Train ego policy based on simulated data

- Then the value discrepancy bound is

$$\left|\eta[\pi^i,\pi^{-i}] - \eta^{\text{branch}}[(\pi_D^1,\hat{\pi}^1),\ldots,(\pi_D^i,\pi^i),\ldots,(\underbrace{\pi_D^n}_{\text{before branch}},\overbrace{\hat{\pi}^n}^{\text{after branch}})]\right|$$

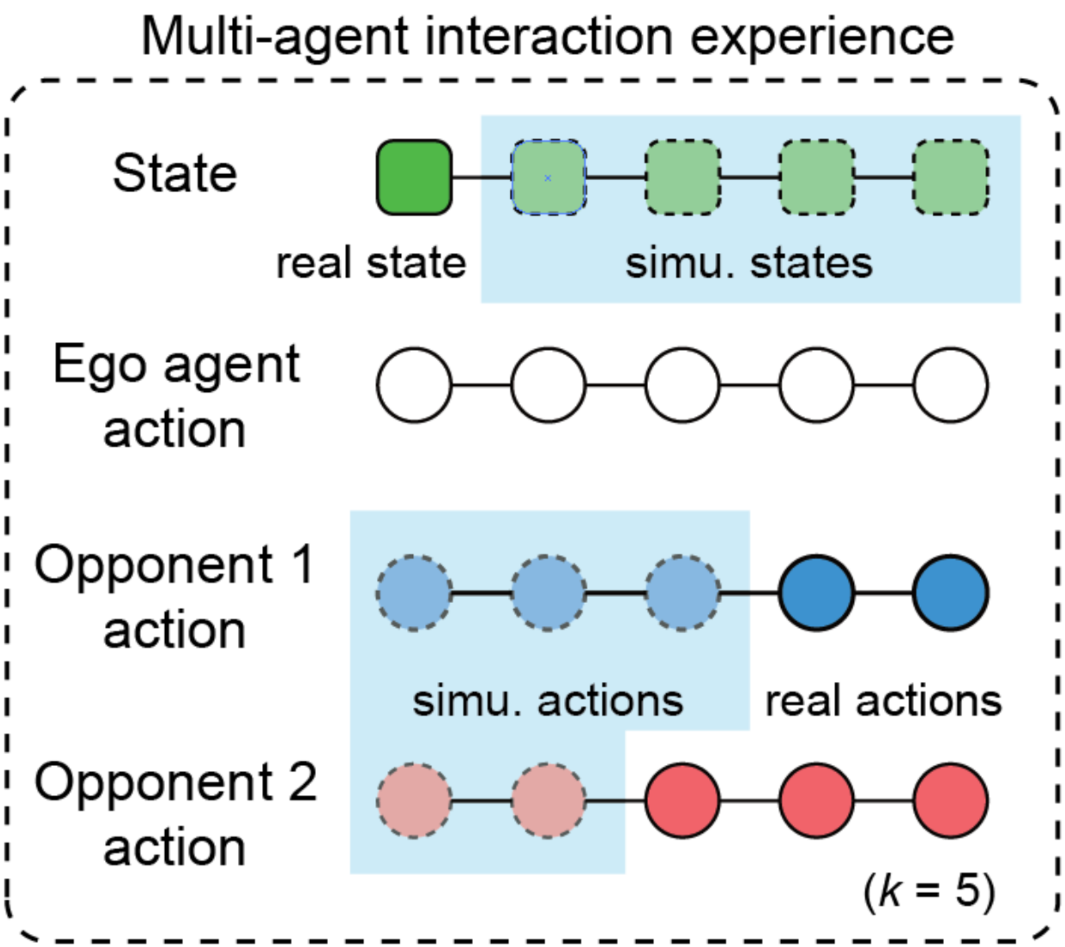
$$\leq 2r_{\max}\Big[\underbrace{k\epsilon'_m + (k+1)\sum_{j\in\{-i\}}\epsilon_{\hat{\pi}}^j}_{\text{model generalization error}} + \underbrace{\gamma^{k+1}\big(\epsilon_\pi^i + \sum_{j\in\{-i\}}\epsilon_\pi^j\big) + \frac{\gamma^{k+1}(\epsilon_\pi^i + \sum_{j\in\{-i\}}\epsilon_\pi^j)}{1-\gamma}}_{\text{policy distribution shift}}\Big]$$

$$= C(\epsilon'_m, \epsilon_\pi^i, \epsilon_\pi^{-i}, \epsilon_{\hat{\pi}}^{-i}, k)$$

different opponent models contribute differently

Weinan Zhang et al. Model-based Multi-agent Policy Optimization with Adaptive Opponent-wise Rollouts. IJCAI 2021.

Algorithm Derived from Theoretic Analysis

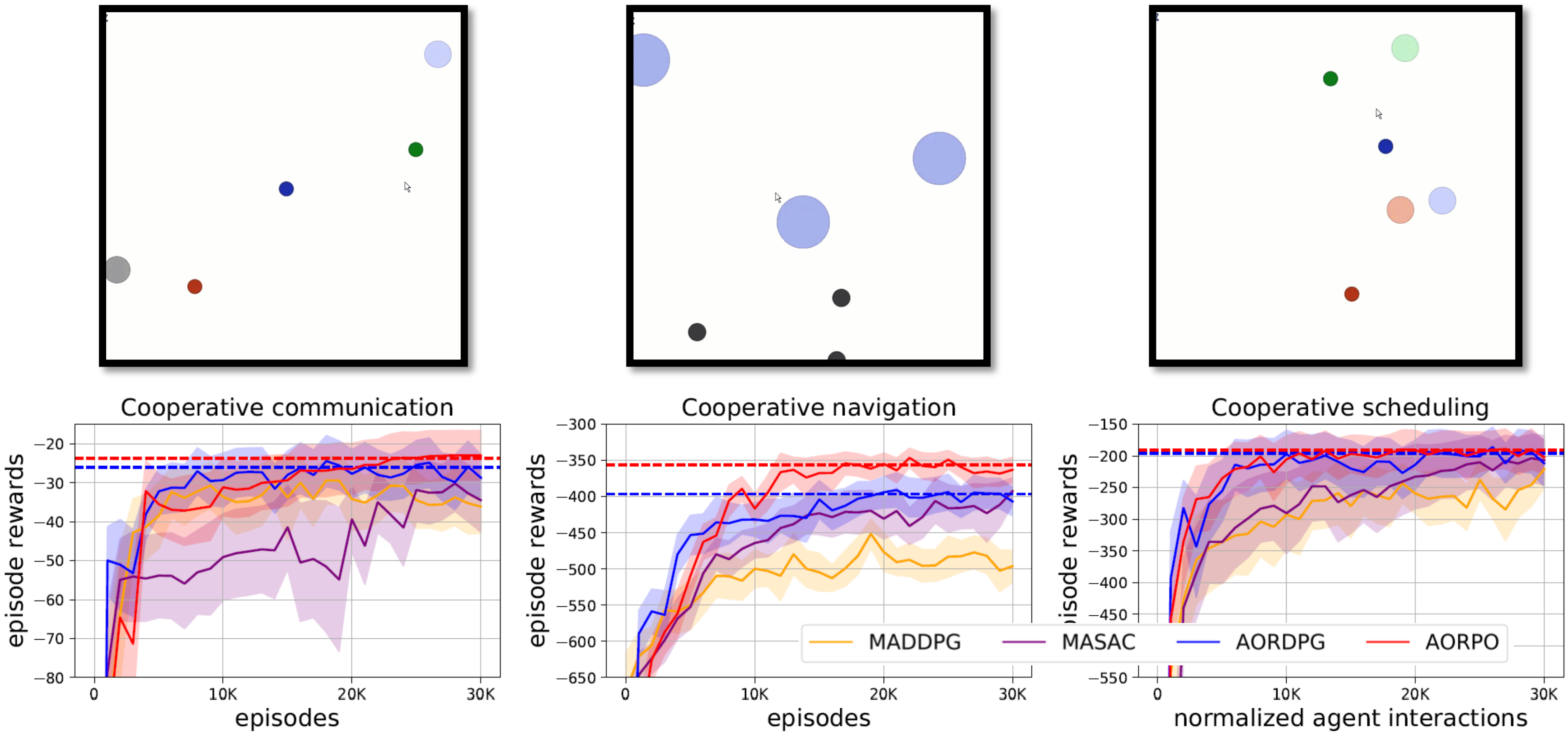


- Environment model starts from real state and then samples the next states
- Ego agent takes actions following its current policy
- Opponent models sample actions based on their errors
  - If the error is large(small), it samples few(more) actions
- Then use real opponent agents to sample further actions

Weinan Zhang et al. Model-based Multi-agent Policy Optimization with Adaptive Opponent-wise Rollouts. IJCAI 2021.

AORPO Experiments

- Multi-Agent Particle Environment (cooperative setting)
  - On sample efficiency, AORPO and AORDPG outperform MASAC and MADDPG respectively, indicating the efficacy of building multi-agent environment model



Complex env. & simple opponents

Simple env. & complex opponents

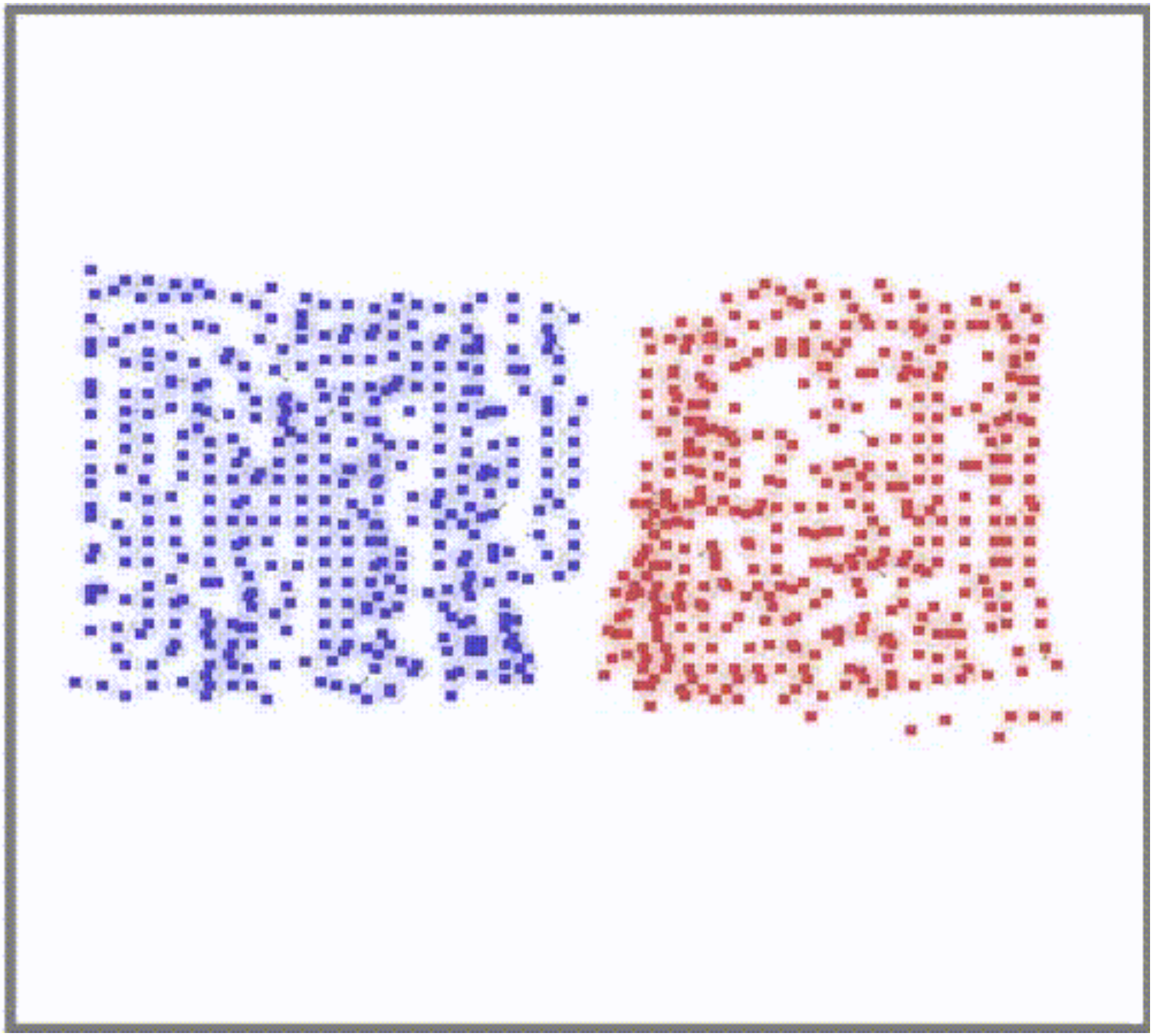
Content

- Introduction to Reinforcement Learning
- Fundamentals of Game Theory
- Multi-Agent Reinforcement Learning
- Many-Agent Reinforcement Learning
  - Algorithms
  - Platforms

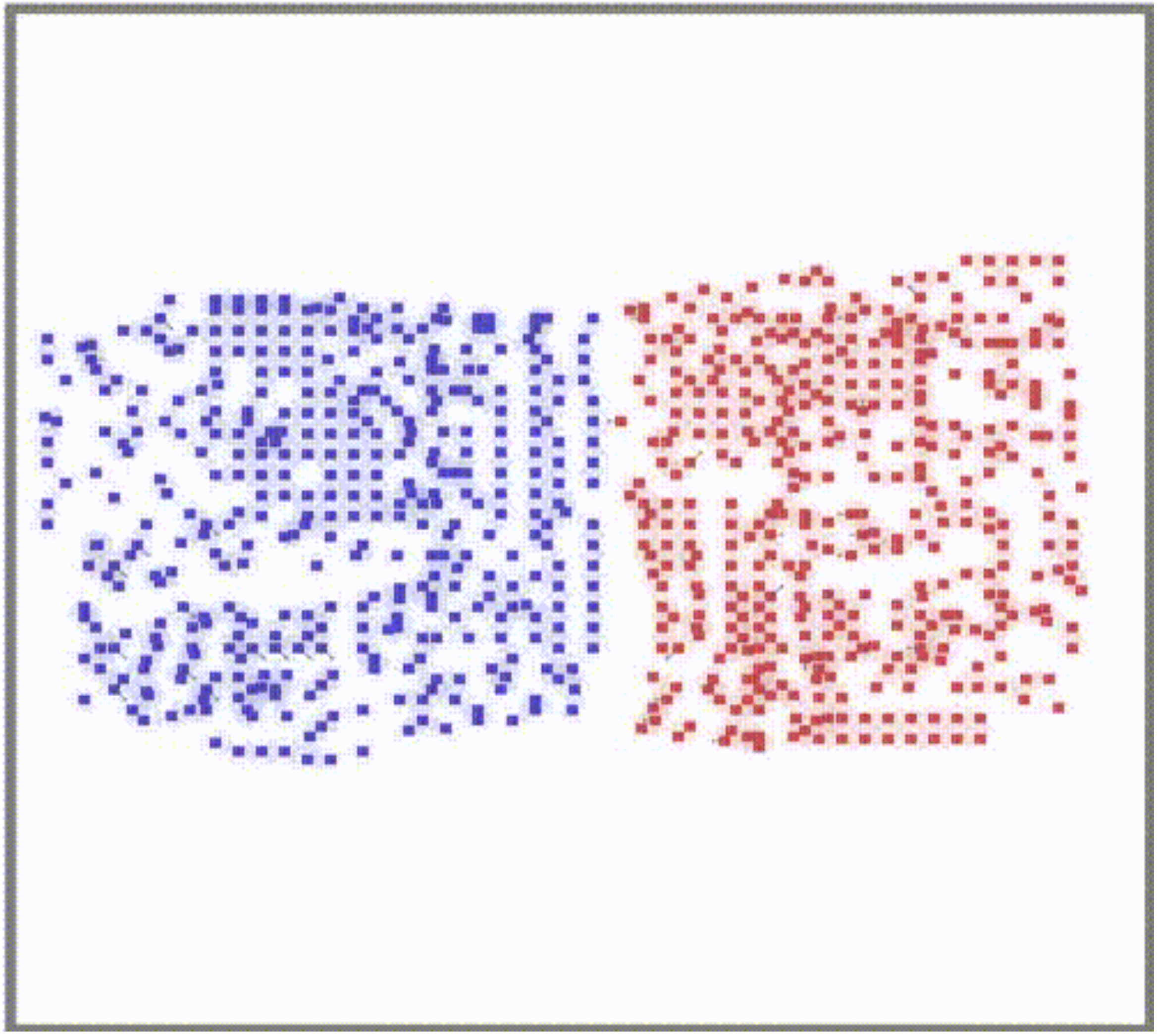
From Multi- to Many-Agent RL

- What will happen when agent number grows?
  - Reward function of agent $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$
  - Transition probability $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Omega(\mathcal{S})$
- Both reward function and state transition probability get exponentially larger
  - More difficult to model
  - The environment is more dynamic and sensitive
  - Need more exploration data
  - More computational resources

MAgent by Geek.AI Battle Game Demo 1



Battle Game Demo 2



Idea 1: Taking Other Agents as A Whole

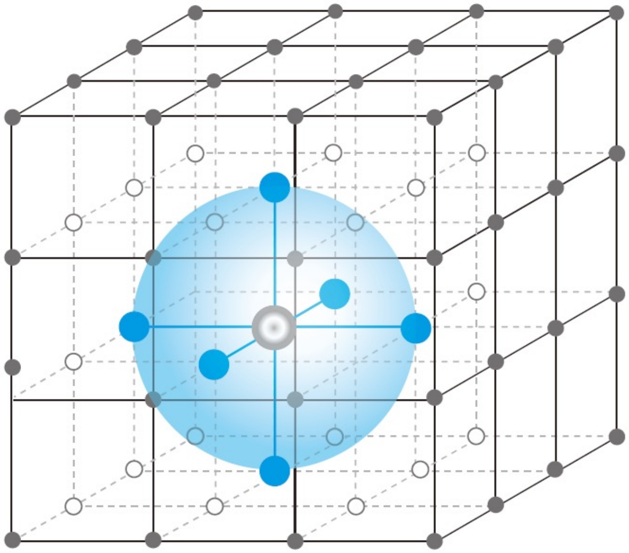
- In some many-body systems, the interaction between an agent and others can be approximated as that between the agent and the “mean agent” of others

Mean Field Multi-Agent RL

- Mean field approximation
  - Approximate the joint action value by factorizing the Q-function into pairwise interactions

$$Q^j(s, \boldsymbol{a}) = \frac{1}{N^j} \sum_{k \in \mathcal{N}(j)} Q^j(s, a^j, a^k)$$

Neighboring agent set of *j*



  - Significantly reduces the global interactions among agents
  - Still preserves global interactions of any agent pair

Yaodong Yang, Weinan Zhang et al. Mean Field Multi-Agent Reinforcement Learning. ICML 2018.

Action Representation

$$Q^j(s, \boldsymbol{a}) = \frac{1}{N^j} \sum_{k \in \mathcal{N}(j)} Q^j(s, a^j, a^k)$$

- Consider discrete action space
  - Action a^j of agent j is one-hot encoded as

$$a^j \triangleq [a^j_1, \ldots, a^j_D]$$

Only one element is 1

  - The mean action based on the neighborhood of j is

$$\bar{a}^j = \frac{1}{N^j} \sum_k a^k$$

  - Thus the action a^k of each neighbor k can be represented as

$$a^k = \bar{a}^j + \delta a^{j,k}$$

mean action

residual

$$\frac{1}{N^j} \sum_k a^{j,k} = 0$$

Residual sum is 0

Mean Field Approximation

- A 2-order Taylor expansion on Q-function

$$a^k = \bar{a}^j + \delta a^{j,k}$$

$$\begin{aligned}
Q^j(s,\boldsymbol{a}) &= \frac{1}{N^j}\sum_k Q^j(s,a^j,a^k) \\
&= \frac{1}{N^j}\sum_k \left[Q^j(s,a^j,\bar{a}^j) + \nabla_{\bar{a}^j}Q^j(s,a^j,\bar{a}^j)\cdot \delta a^{j,k} + \frac{1}{2}\delta a^{j,k}\cdot \nabla^2_{\tilde{a}^{j,k}}Q^j(s,a^j,\tilde{a}^{j,k})\cdot \delta a^{j,k}\right] \\
&= Q^j(s,a^j,\bar{a}^j) + \nabla_{\bar{a}^j}Q^j(s,a^j,\bar{a}^j)\cdot \frac{1}{N^j}\sum_k \delta a^{j,k} + \frac{1}{2N^j}\sum_k \delta a^{j,k}\cdot \nabla^2_{\tilde{a}^{j,k}}Q^j(s,a^j,\tilde{a}^{j,k})\cdot \delta a^{j,k} \\
&= Q^j(s,a^j,\bar{a}^j) + \frac{1}{2N^j}\sum_k R^j_{s,a^j}(a^k) \\
&\approx Q^j(s,a^j,\bar{a}^j)
\end{aligned}$$

External random signal for agent *j*

$$R^j_{s,a^j}(a^k) \triangleq \delta a^{j,k}\cdot \nabla^2_{\tilde{a}^{j,k}}Q^j(s,a^j,\tilde{a}^{j,k})\cdot \delta a^{j,k}$$

$$\tilde{a}^{j,k} = \bar{a}^j + \epsilon^{j,k}\delta a^{j,k}$$

Q-function model the interaction between the agent's action and the mean action

Mean Field Q-Learning

- A softmax MF-Q policy

$$\pi_t^j(a^j|s,\bar{a}^j) = \frac{\exp\big(-\beta Q_t^j(s,a^j,\bar{a}^j)\big)}{\sum_{a^{j'}\in\mathcal{A}^j}\exp\big(-\beta Q_t^j(s,a^{j'},\bar{a}^j)\big)}$$

- Given an experience $\langle s, \boldsymbol{a}, \boldsymbol{r}, s', \bar{\boldsymbol{a}}\rangle$ sampled from replay buffer
  - Sample the next action a_-^j from $Q_{\phi_-^j}$
  - Set $y^j = r^j + \gamma\, Q_{\phi_-^j}(s', a_-^j, \bar{a}^j)$
  - Update Q function with the loss function

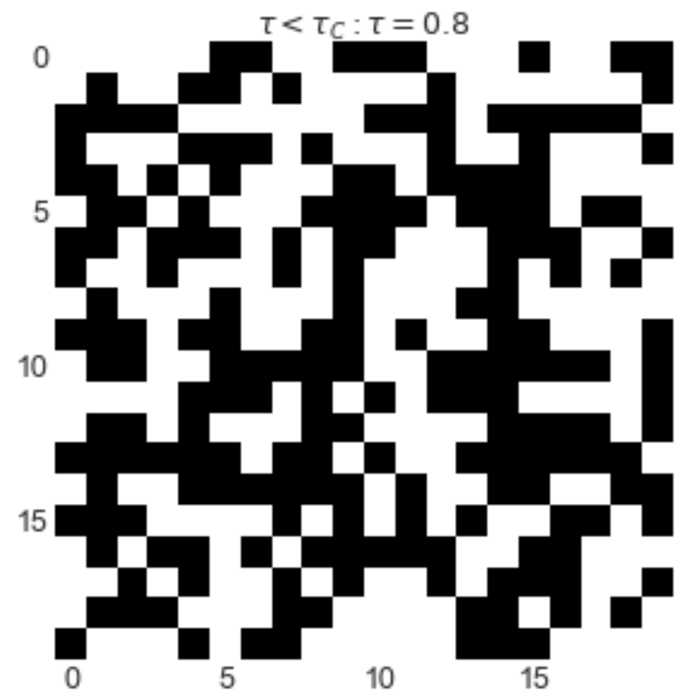
$$\mathcal{L}(\phi^j) = \big(y^j - Q_{\phi^j}(s^j, a^j, \bar{a}^j)\big)^2$$

Experiment of Ising Model

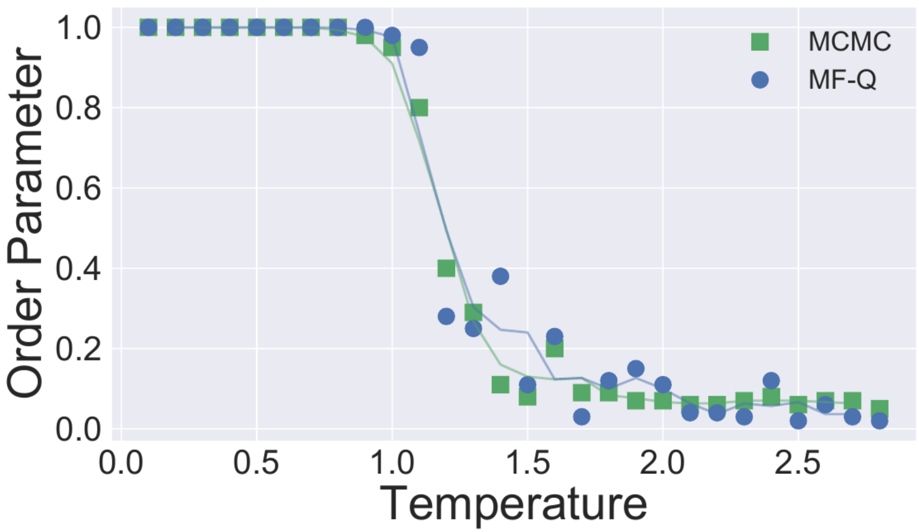
- Each spin is an agent to decide up or down (action)
- Measure: order parameter

$$\xi = \frac{|N_\uparrow - N_\downarrow|}{N}$$

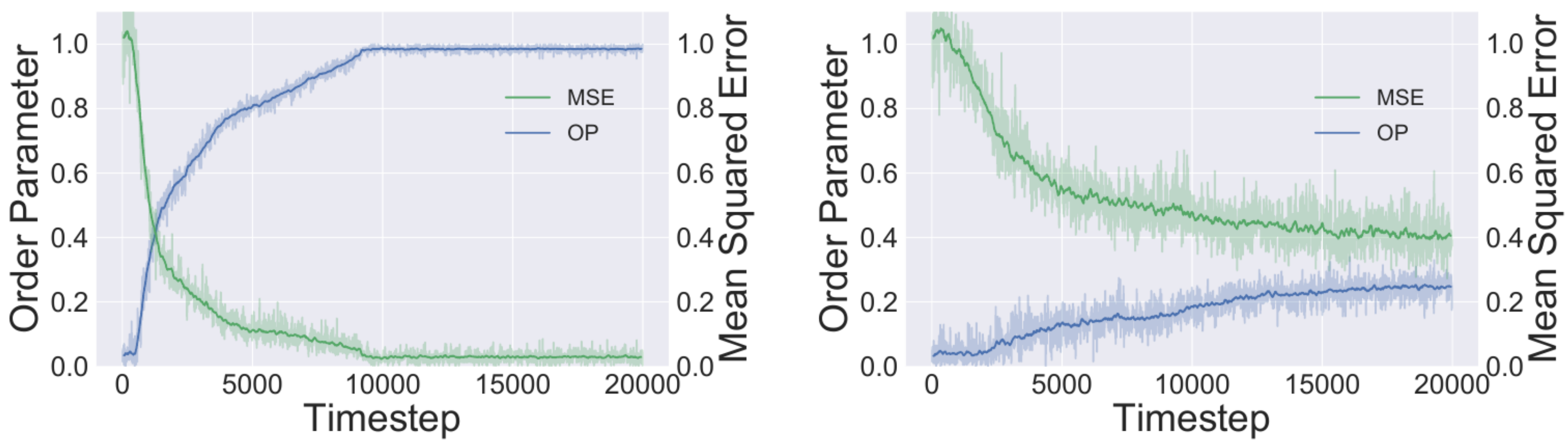
  - The closer OP is to 1, the more orderly the system is.



Experiment Performance IM

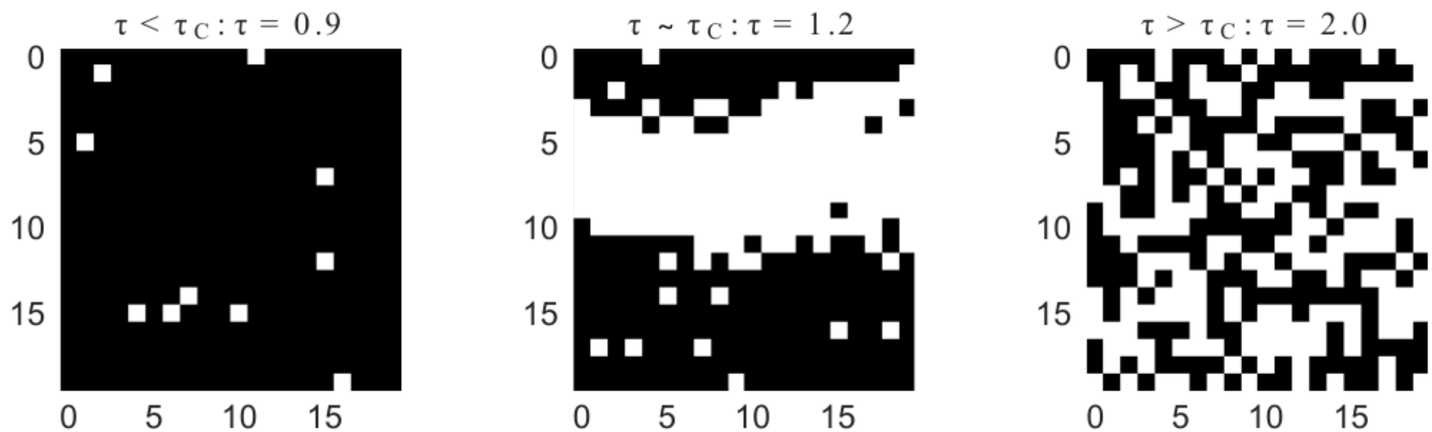


- Ground truth: MCMC simulation
- Goal: MF-Q learns with the similar behavior as MCMC, which we observed

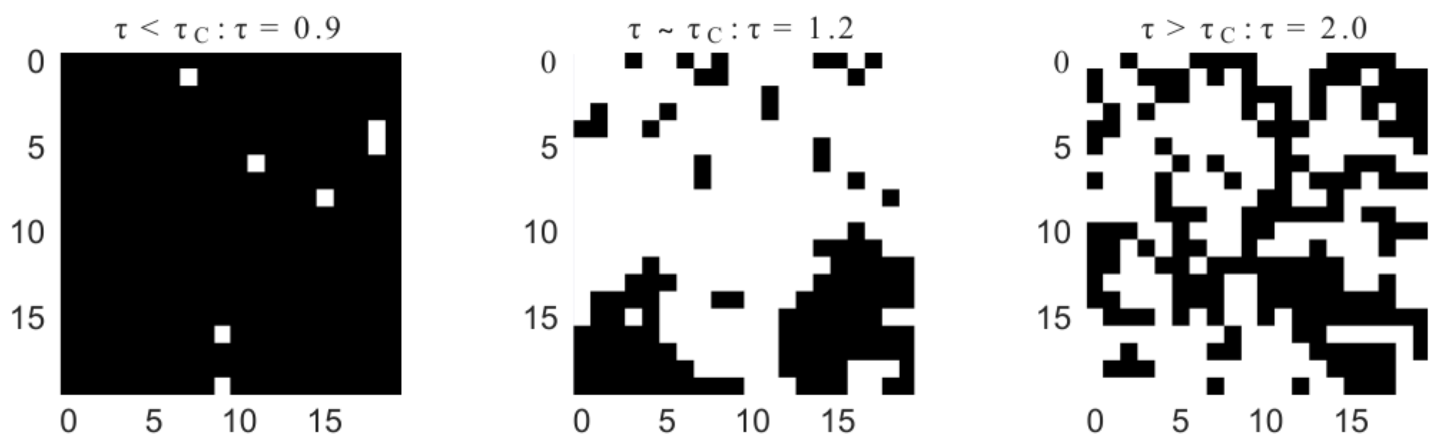


(a) $\tau = 0.8$ (b) $\tau = 1.2$

Experiment Performance IM



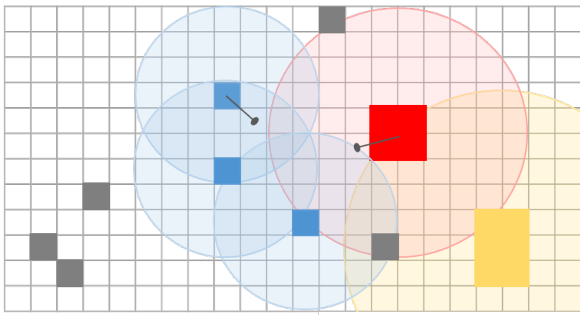
(a) MF-Q



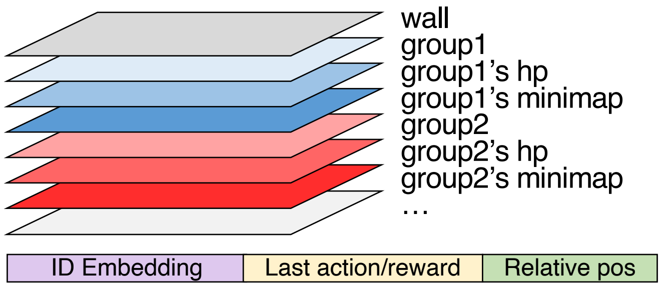
(b) MCMC

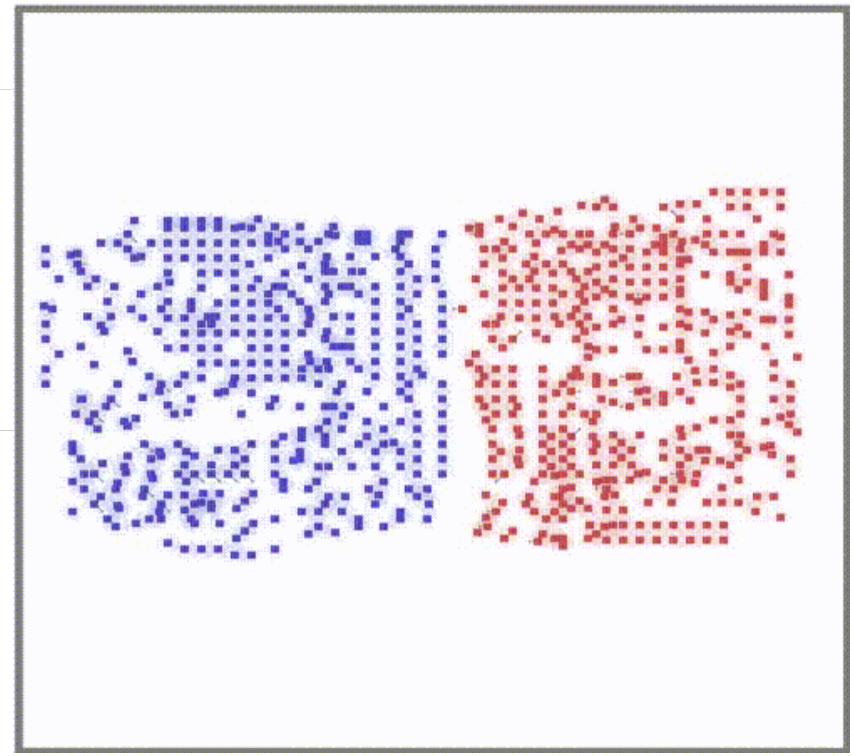
Experiment: Battle

Grid World

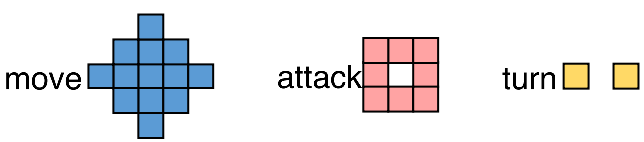


Observation Space





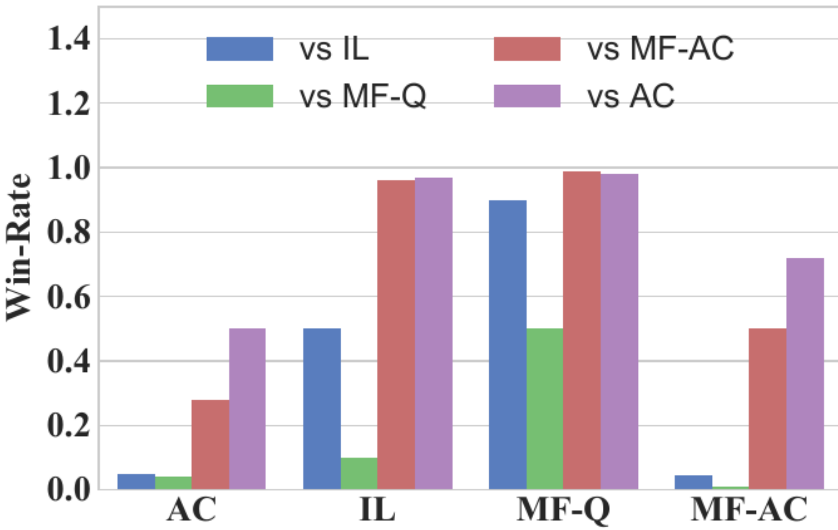
Action Space



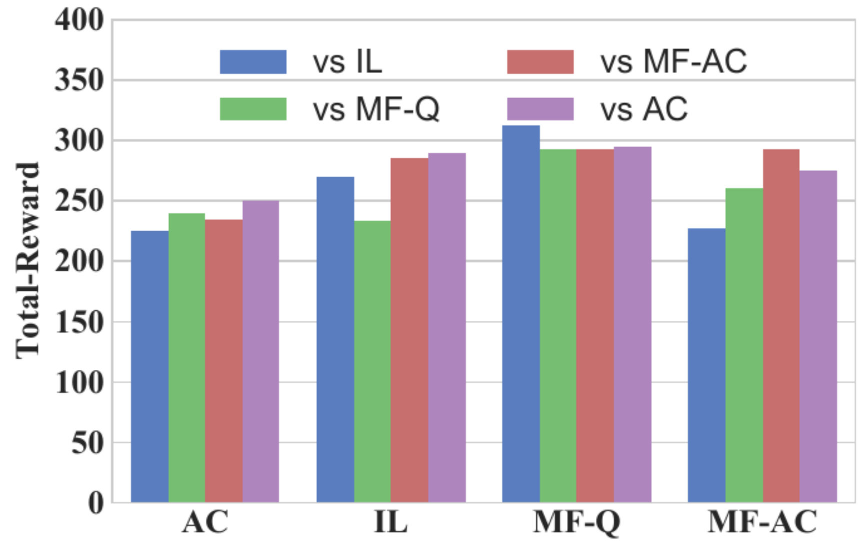
Supported by MAgent by Geek.AI

Lianmin Zheng, Weinan Zhang et al. "MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence." *NIPS 2017.*

Experiment Performance Battle



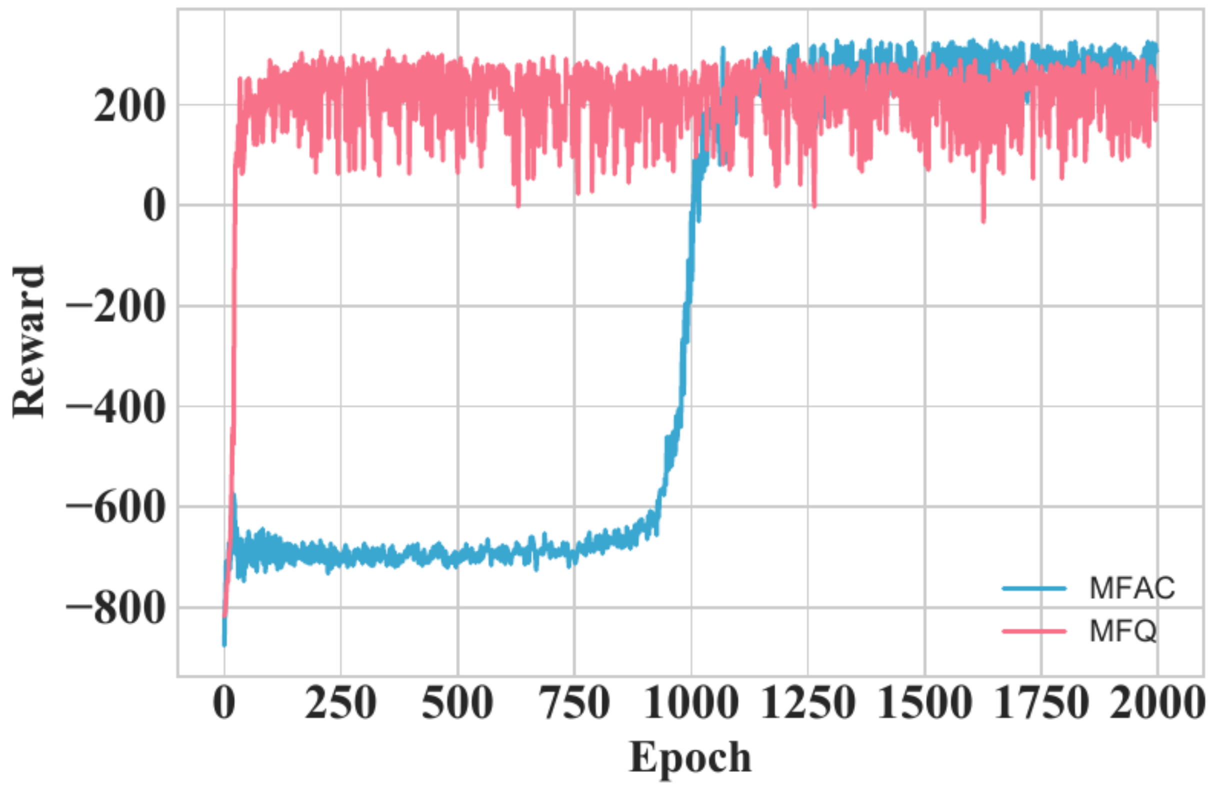
(a) Average wining rate.



(b) Average total reward.

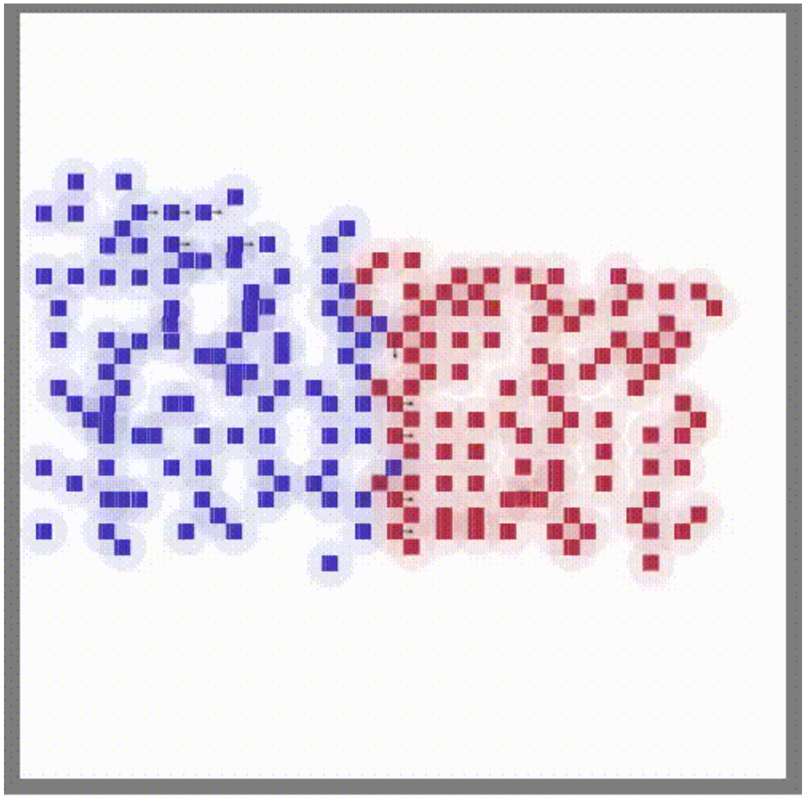
- For 64 vs 64 battle, MF-Q works the best among all compared models
- MF-AC may not work that well particularly when the agent number is large

Experiment Performance Battle



- MF-Q has a fast convergence property
- MF-AC has a phase changing point

Case Study of MF-Q vs. IL

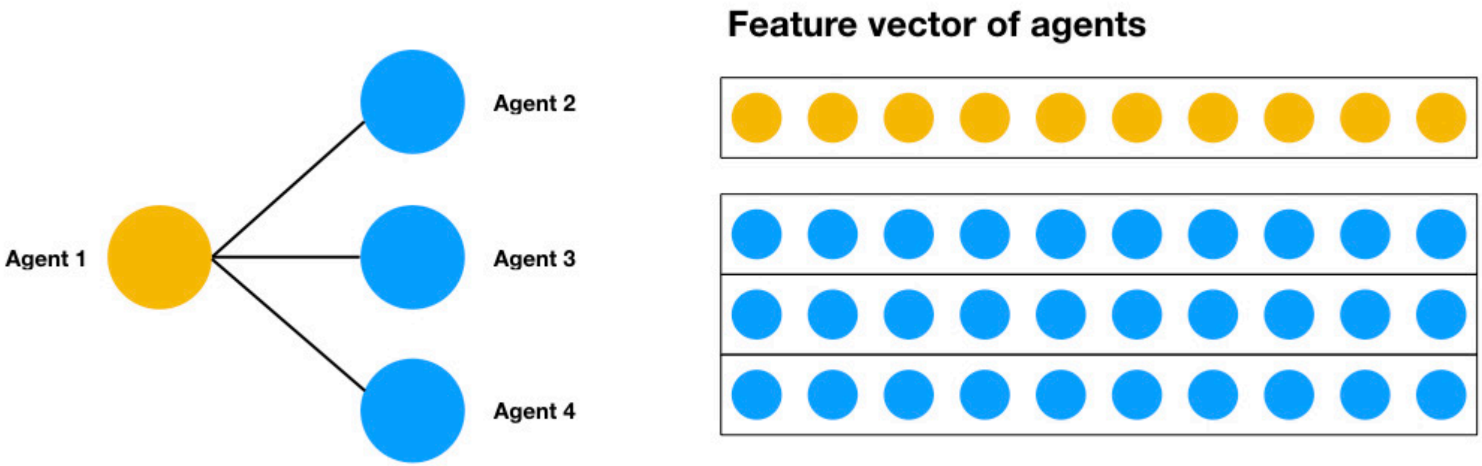


- Blue: MF-Q
- Red: IL

- MF-Q presents a go-around-and-besiege strategy

- MF-Q agents are more consistent with neighbors

Idea2: Factorization Machine



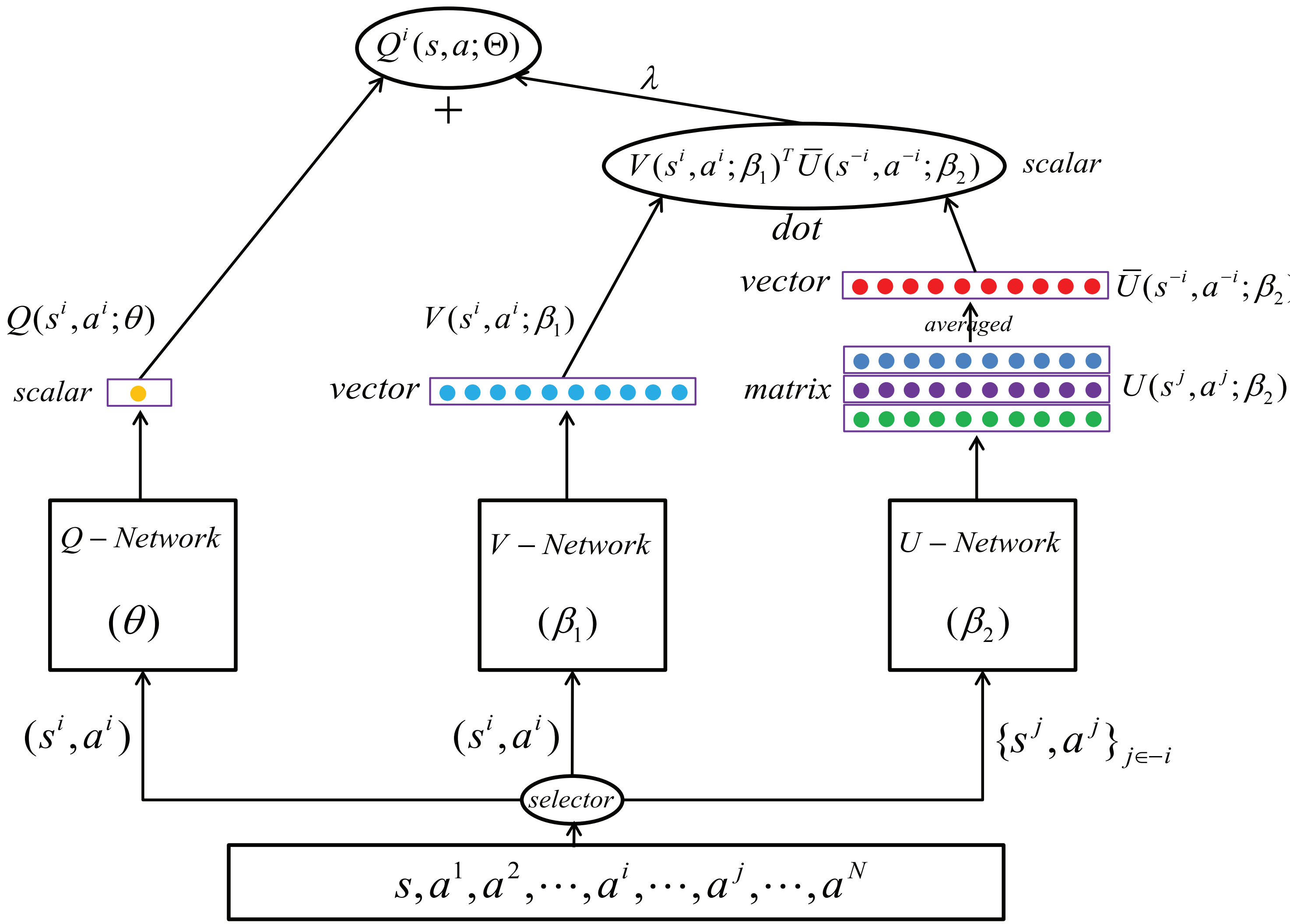
- Computing the Q-value with the independent Q-function and the interactive ingredients inspired from factorization machine

Factorized Q-learning

- A composite deep neural network architecture whose components share the model parameters among all the agents within the same group
  - Reduce the model complexity
  - Still preserves global interactions of any agent pair
  - Accelerate the learning process.

Ming Zhou, Weinan Zhang et al. Factorized Q-Learning for Large-Scale Multi-Agent Systems. DAI 2019.

Factorized Q-learning



Ming Zhou, Weinan Zhang et al. Factorized Q-Learning for Large-Scale Multi-Agent Systems. DAI 2019.

Factorized Q-learning

- Q-Network: Denote the i-th agent's value function

$$Q(s^i, a^i, \theta)$$

- V-Network & U-Network
  - The outputs of V and U denote the feature vectors of focused agent and other agents, respectively. And the dot product of U and V denotes the interactive ingredients of the focused i-th agent and other j-th agents

$$V(s^i, a^i; \beta_1)^T \sum_{j \in -i} U(s^j, a^j; \beta_2)$$

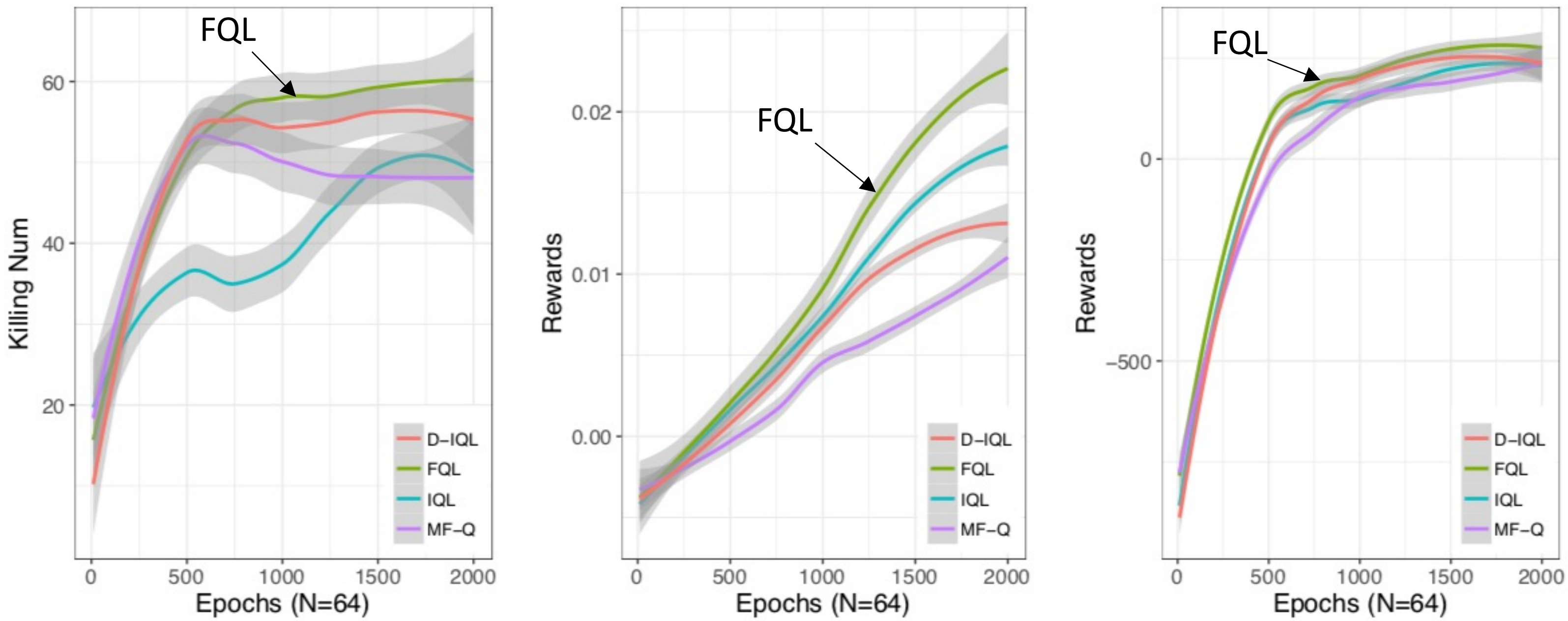
Factorized Q-learning

- We redefine the Q-function for the high-order tensor relationship between states and actions as follows

$$
\begin{aligned}
Q^i(s, a^1, a^2, \cdots, a^N; \Theta) &\equiv Q^i(s, a^i, a^{-i}; \Theta) \\
&\approx Q^i(s, a^i; \theta) + \lambda^o \sum_{j \in -i} V(s, a^i; \beta_1)^T U(s, a^j; \beta_2) \\
&\approx Q(s^i, a^i; \theta) + \lambda^o \cdot V(s^i, a^i; \beta_1)^T \sum_{j \in -i} U(s^j, a^j; \beta_2) \\
&= Q(s^i, a^i; \theta) + \lambda \cdot V(s^i, a^i; \beta_1)^T \left(\frac{1}{N-1}\right) \sum_{j \in -i} U(s^j, a^j; \beta_2) \\
&= Q(s^i, a^i; \theta) + \lambda \cdot V(s^i, a^i; \beta_1)^T \overline{U}(s^{-i}, a^{-i}; \beta_2),
\end{aligned}
$$

$\bar{U}(s^{-i}, a^{-i}, \beta^2) = \frac{1}{N-1} \sum_{j \in -i} U(s^j, a^j, \beta_2)$ implies the equivalent force in place of complex interactions

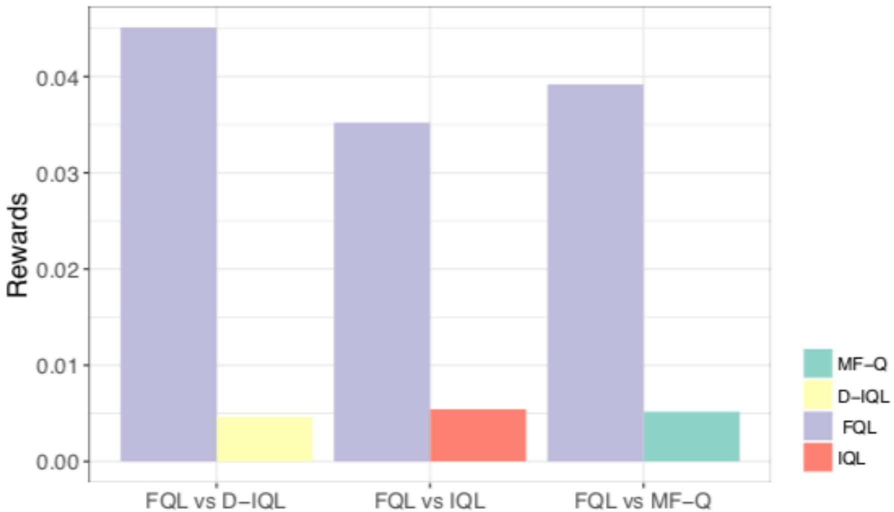
Experiment Performance Battle



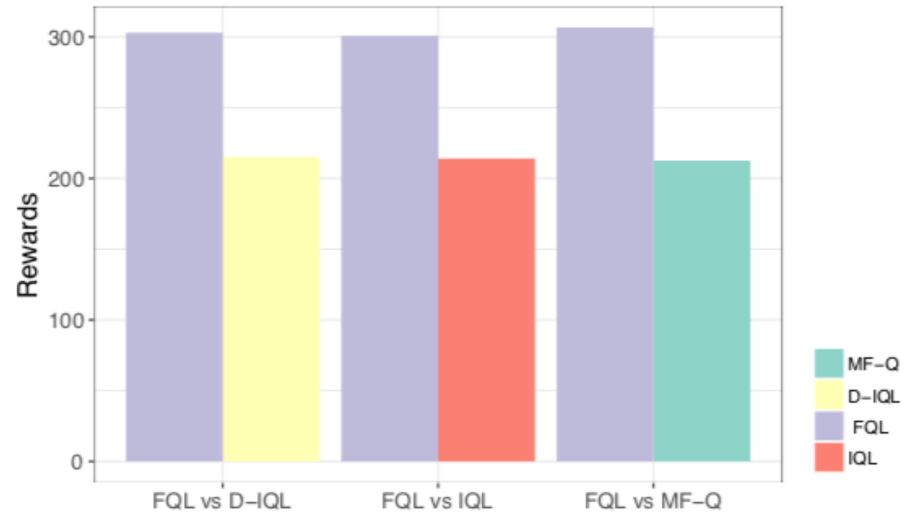
(a) Killing Index (b) Mean-Rewards Index (c) Total Rewards Index

- 3 self-play trainining curves. The Killing Index shows the ability of killing enemies, the Mean-Rewards Index shows the rewards of every agent in every step, and the Total Rewards Index shows the ability of gaining rewards in an episode.

Experiment Performance Battle



(a) Mean-Rewards Index



(b) Total Rewards Index

- The battle results between FQL and other three competitors indicate the effectiveness of FQL

Content

- Introduction to Reinforcement Learning
- Fundamentals of Game Theory
- Multi-Agent Reinforcement Learning
- Many-Agent Reinforcement Learning
  - Algorithms
  - Platforms

From Multi- to Many-Agent RL

- What will happen when agent number grows?
  - Reward function of agent $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$
  - Transition probability $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \Omega(\mathcal{S})$

- Both reward function and state transition probability get exponentially larger
  - More difficult to model
  - The environment is more dynamic and sensitive
  - Need more exploration data
  - More computational resources

Key Factors for Successful MARL

- Computation: High computational resource for reinforcement learning
- Data: a huge amount of data for training the models
- Environment: a low-cost environment for RL agents to interact with
- Solution: an effective simulator

What accounts for an effective simulator?

High Efficiency

- Interact with multiple agents in a high speed
- Multi-thread and multi-machine deployment

High Reality

- The simulation results should be as close to reality as possible
- Match in both micro and macro levels

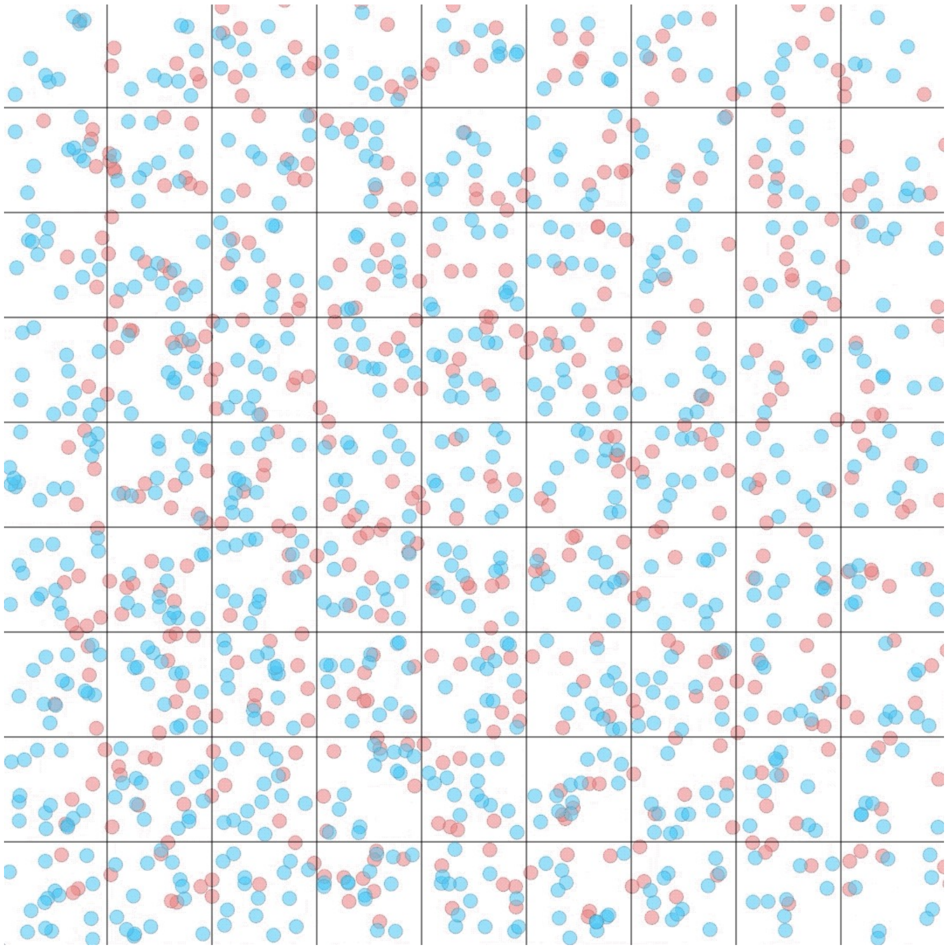
Extendibility

- Flexible to adapt to new tasks with little effort

Interaction

- Easy to visualize and friendly for human interaction

MARL Case: Online Taxi Order Dispatch



Blue points: orders

Red points: taxis

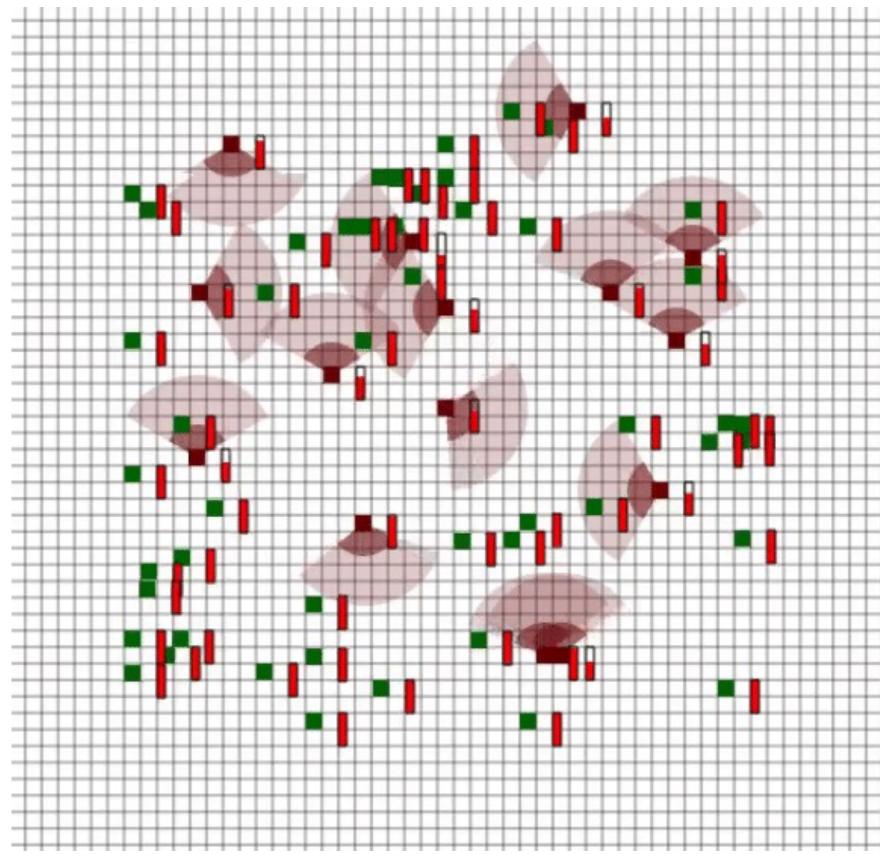
Ming Zhou, Weinan Zhang. Multi-Agent Reinforcement Learning for Order Dispatching via Order-Vehicle Distribution Matching. Working paper.

Featured Simulators

- Discrete world: MAgent
  - https://github.com/geek-ai/MAgent
- Continuous world: Cityflow
  - https://github.com/cityflow-project/CityFlow/
- Self-driving cars: SMARTS
  - https://github.com/huawei-noah/SMARTS

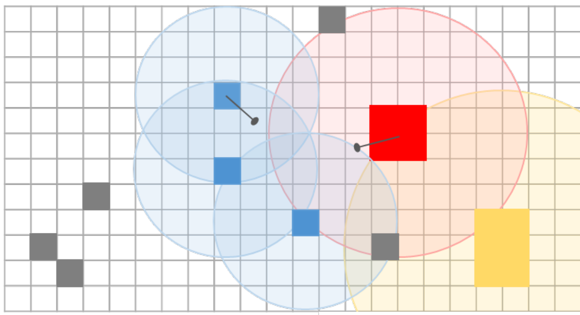
The Challenges

- High demand of computation
  - Large scale computation in training, inference and simulation
- Scalable and dynamic solution
  - The number of agents is highly dynamic. Agents can enter and exit
- Complicated interaction
  - It is hard to exactly model the interaction among agents
- Visualization

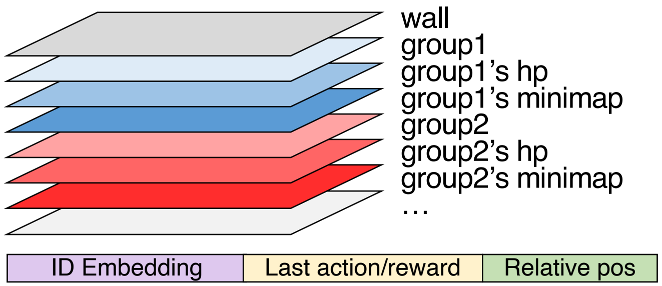


The Challenges

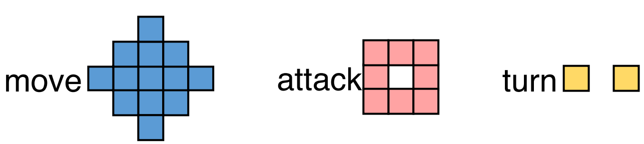
Grid World

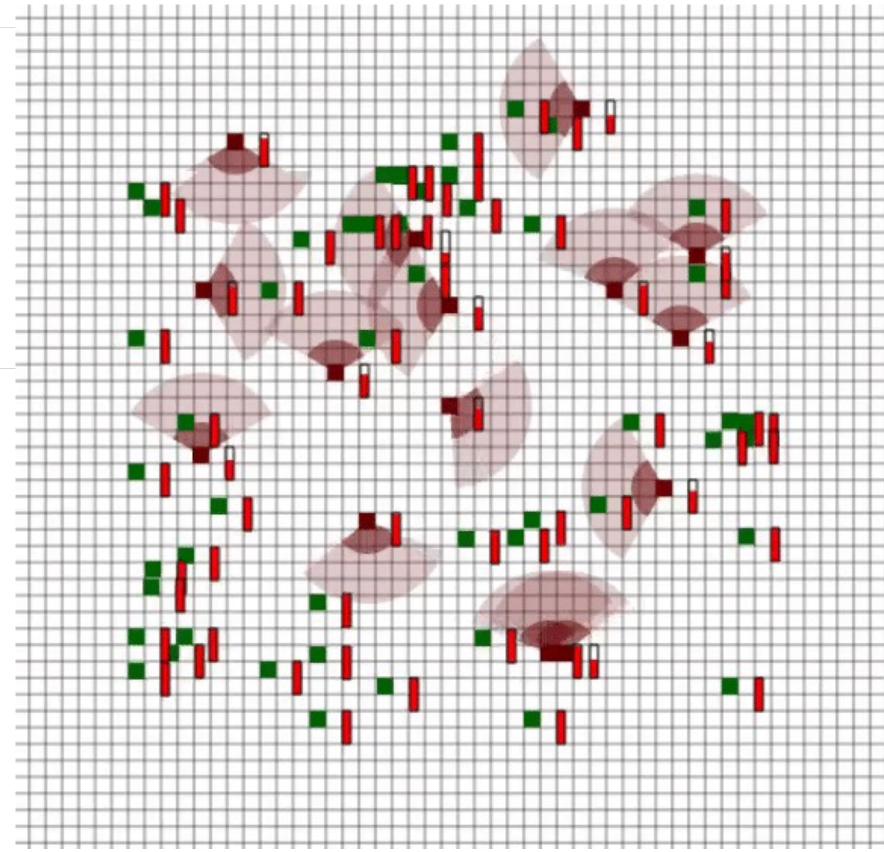


Observation Space



Action Space



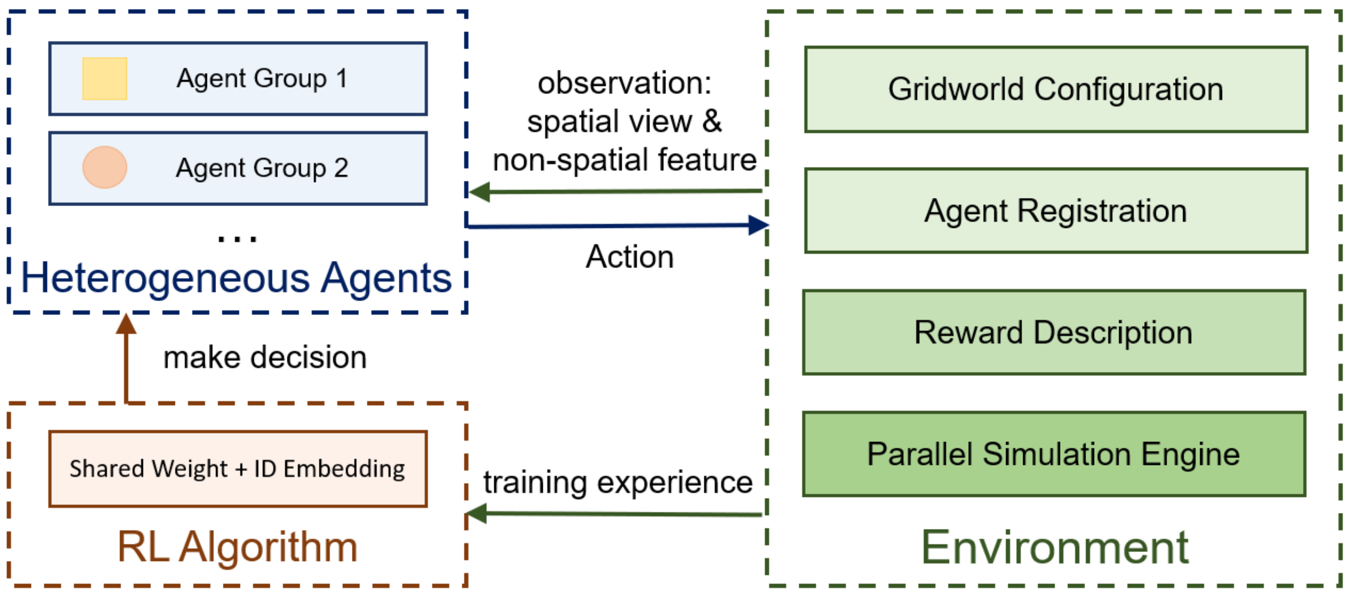


Other MARL Platforms

Platform	Number of agents	Learning Interface
OpenAI Gym	< 10	√
Malmo	< 1000	√
Starcraft Learning Environment	< 2000	√
Arcade Learning Environment	< 10	√
NetLogo	~ 1000,000	×
MAgent	**~ 1,000,000**	√

Decentralized MARL

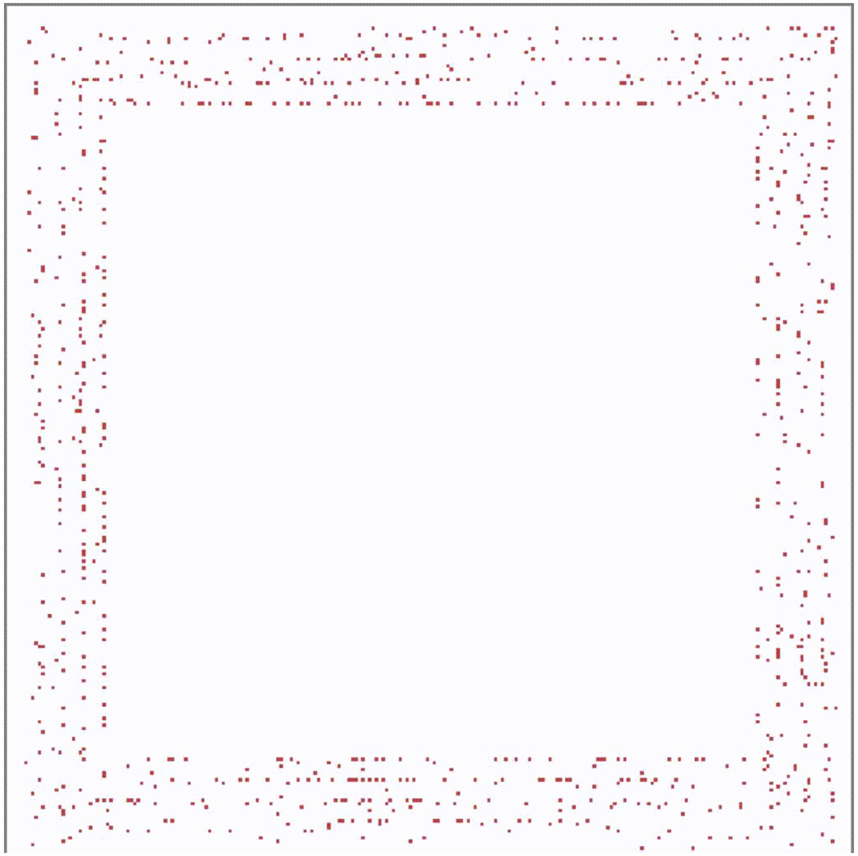
- When agent number is too large to maintain a centralized meta-agent for controlling
  - Sharing Q-network for scalability
  - Agent ID for personalization



$$Q(s_t^i, a_t^i) \leftarrow Q(s_t^i, a_t^i) + \alpha[r_t^i + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}^i, a') - Q(s_t^i, a_t^i)]$$

Use Case: Many-Agent Interactions

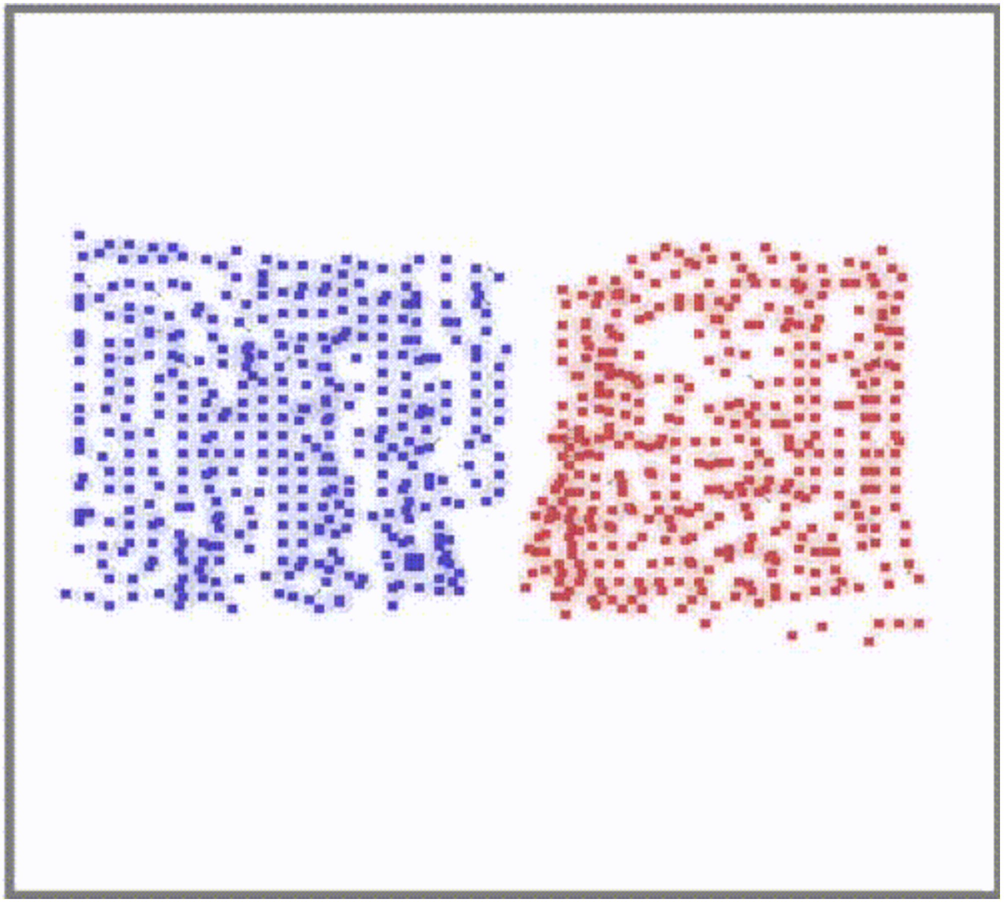
- MAgent game: aligning



Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

Use Case: Many-Agent Interactions

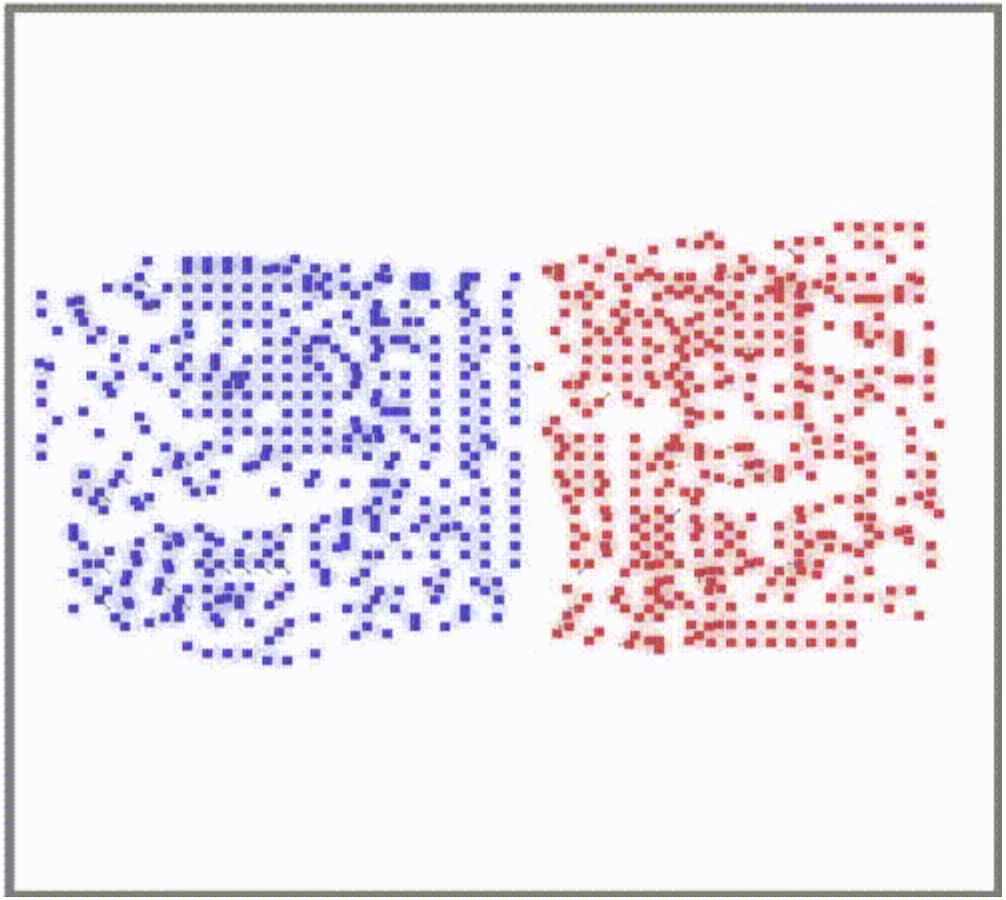
- MAgent game: battle



Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

Use Case: Many-Agent Interactions

- MAgent game: battle



Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

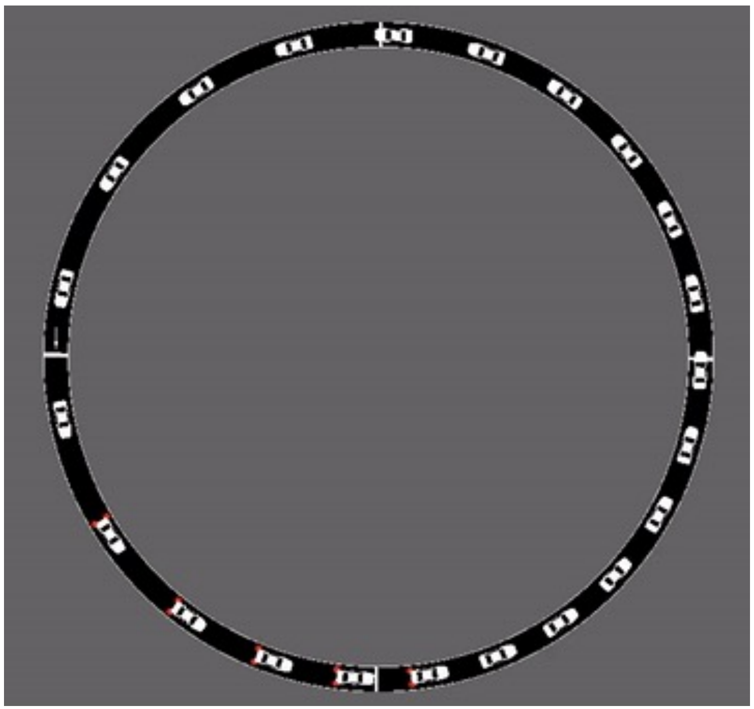
Use Case: Many-Agent Interactions

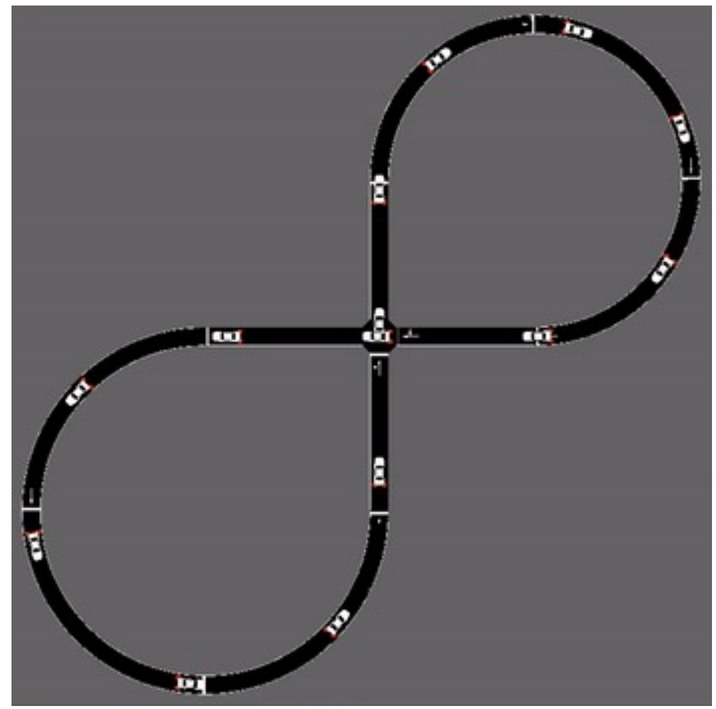
- MAgent game: city simulation

- Designing
  - Car routing policy
  - Traffic light controller
  - Fleet management & taxi dispatch

Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

A Continuous-World Simulator is Necessary to City Traffic Simulation





For example, traffic jams are caused by micro-scale acceleration and deceleration.

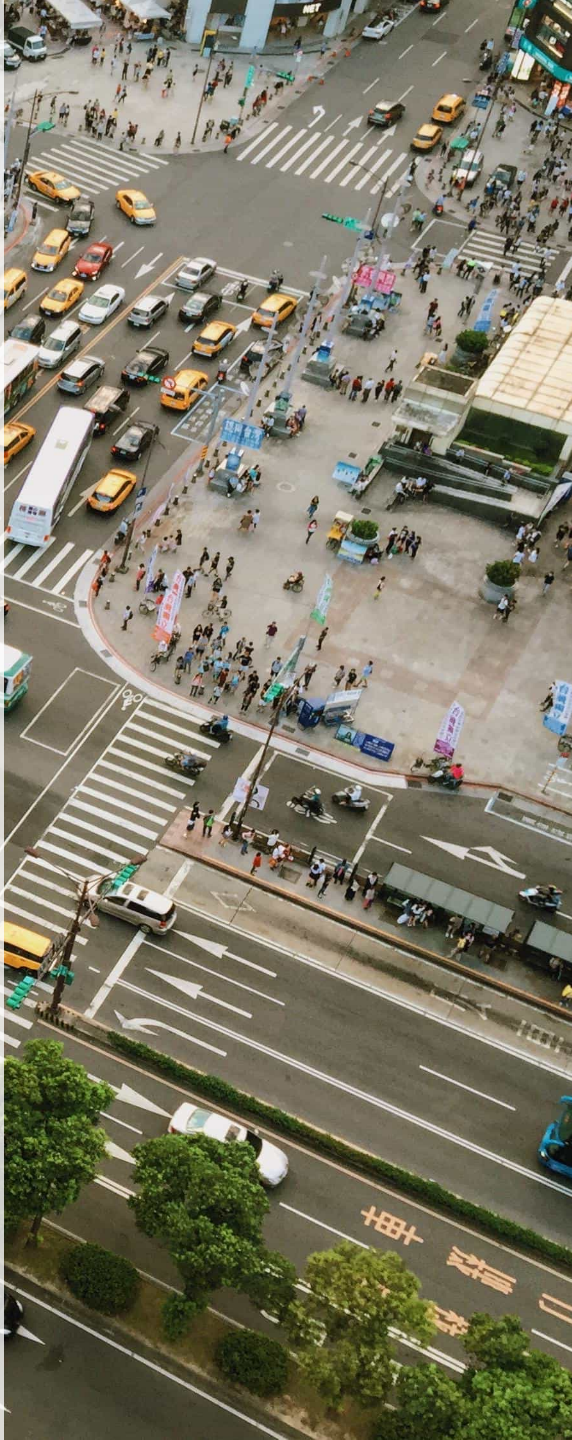
Cathy Wu et al. Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control. 2017.

Featured Simulators

- Discrete world: MAgent
  - https://github.com/geek-ai/MAgent
- Continuous world: Cityflow
  - https://github.com/cityflow-project/CityFlow/
- Self-driving cars: SMARTS
  - https://github.com/huawei-noah/SMARTS

Why is Simulator a Necessity?

- Large number of training samples required
- High trial and error cost
- Not possible in real world environment
- Even more problematic in city scenario

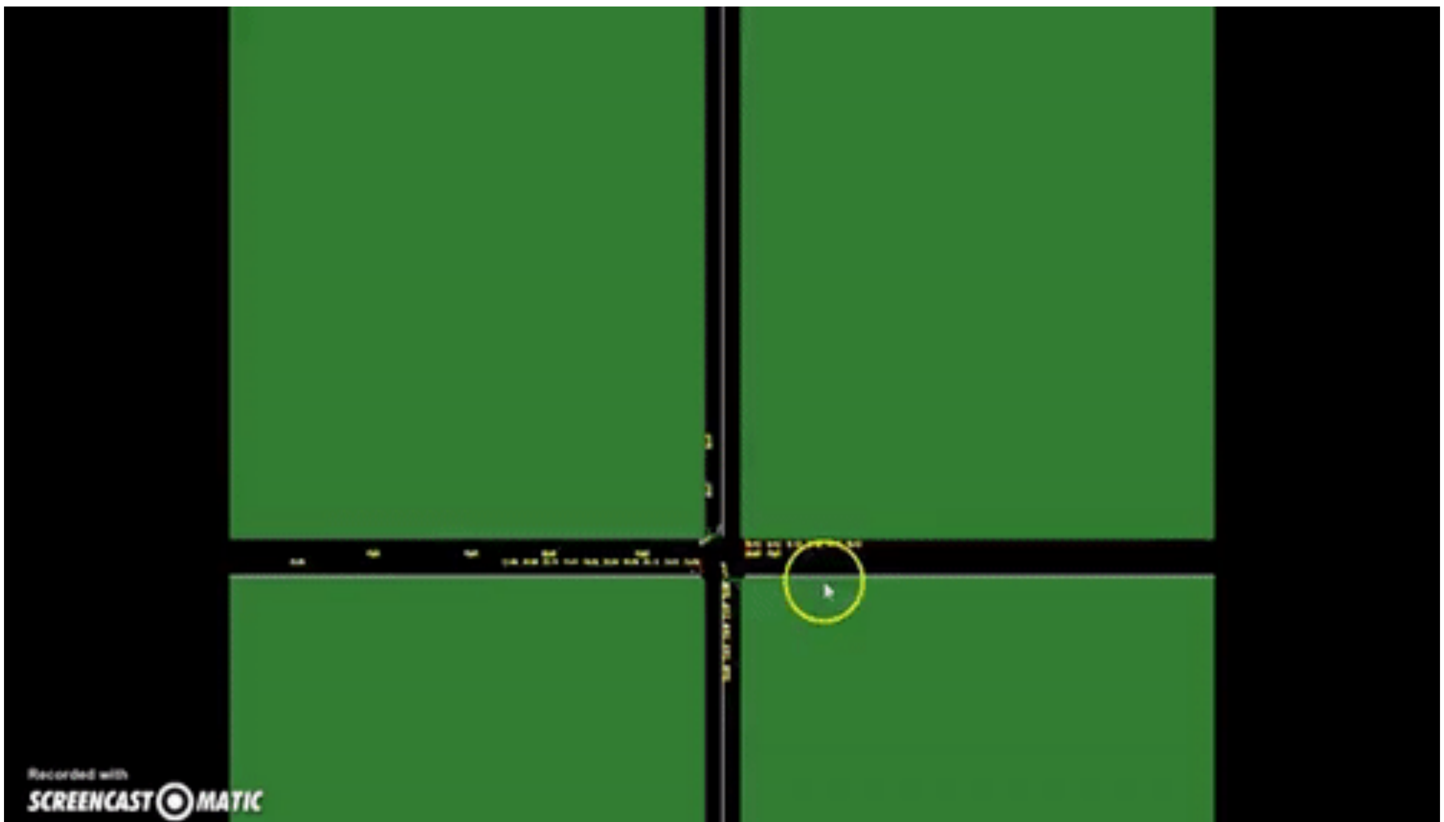


Current City Simulator

- Focus on traffic scenario
  - Commercial:
    - VISSIM
    - AIMSUN.NEXT
  - Open source:
    - Most Popular: SUMO
    - Ours: CityFlow
- Microscopic Simulator
  - Simulate the movement of each single vehicle or object

SUMO

- **S**imulation of **U**rban **Mo**bility
- Institute of Transportation Systems @ German Aerospace Center
- Project starts from 2000



https://gfycat.com/hopefulsolidamericancreamdraft

CityFlow

https://github.com/cityflow-project/CityFlow

- CityFlow is city simulator particularly focused on speed and scale

Parallel Computing

Parallel Computing speeds up 4x with 8 core CPU

Car Following

New Car following algorithm designed by us is much faster than SUMO

Roadnet Design

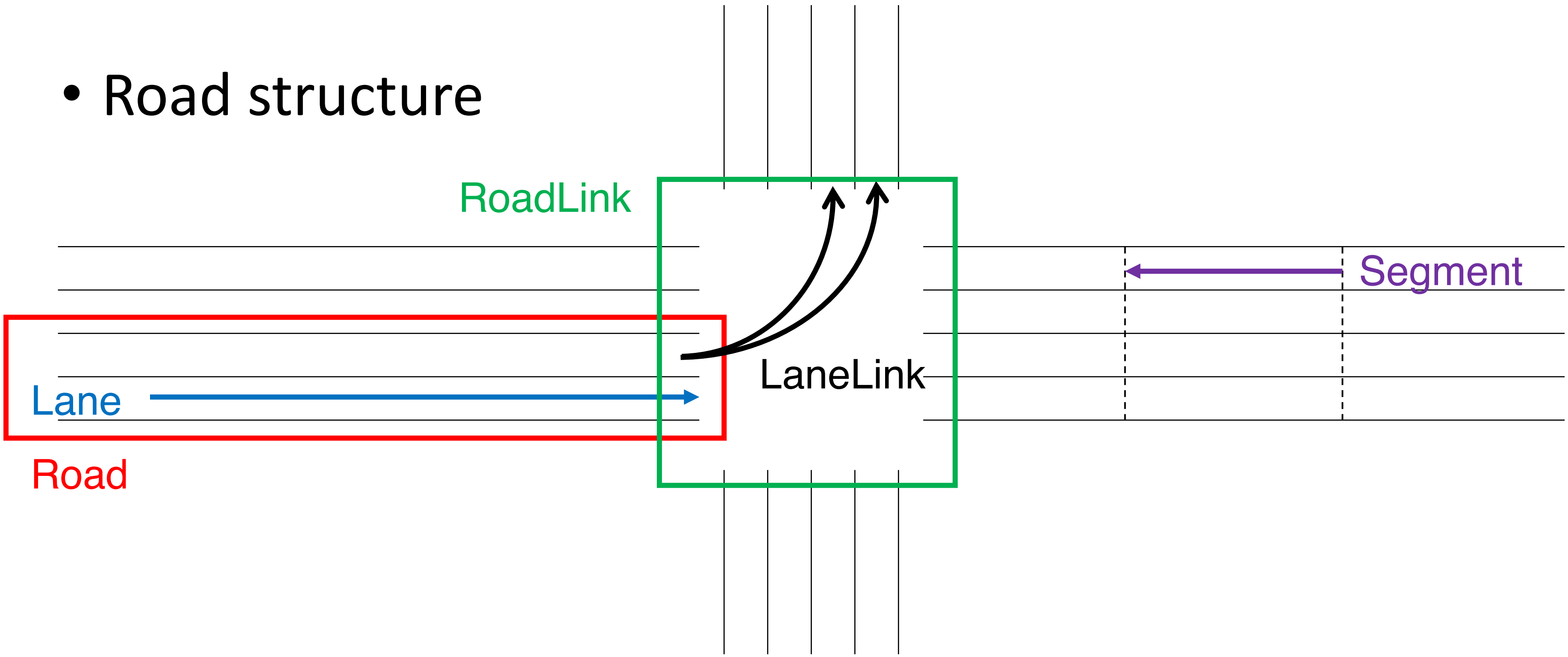
Our hierarchical roadnet structure serves for car following model

Excellent Implement

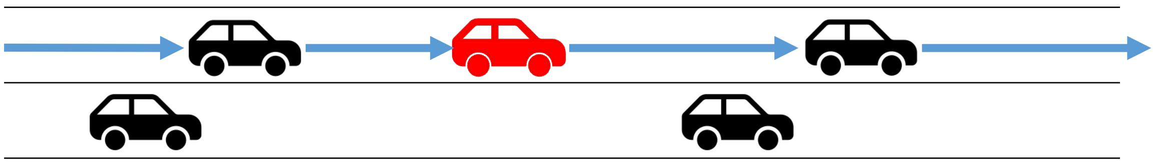
Improvement of fundamental code speeds up 2x with same algorithm

CityFlow Design

- Road structure



- Car flow structure: linked list

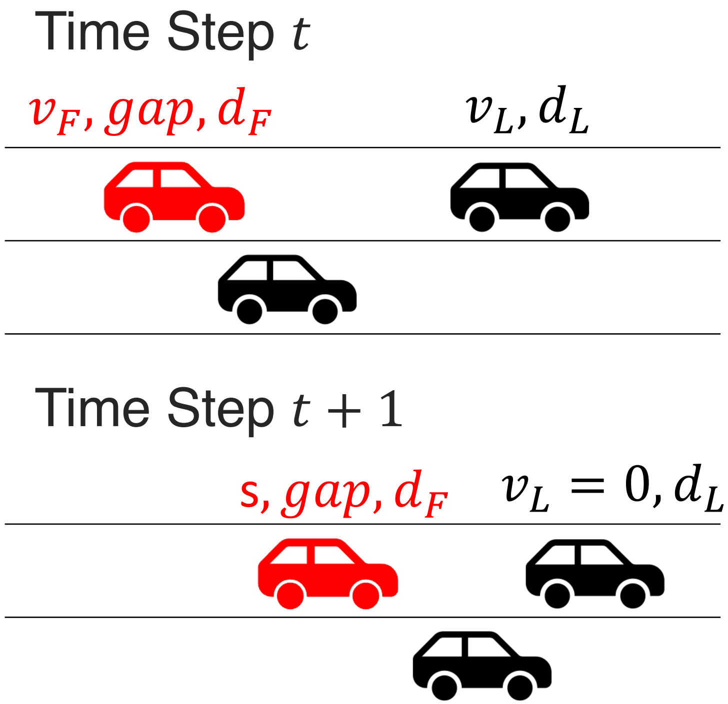


CityFlow Design

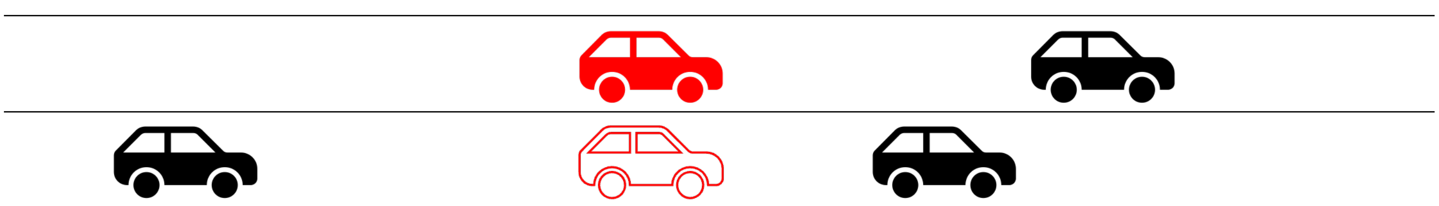
- Car following model

no-collision-speed s:

$$\frac{1}{2d_F}s^2 + \frac{1}{2}interval \cdot s = gap + \frac{v_L^2}{2d_L} - \frac{1}{2}interval \cdot v_F$$



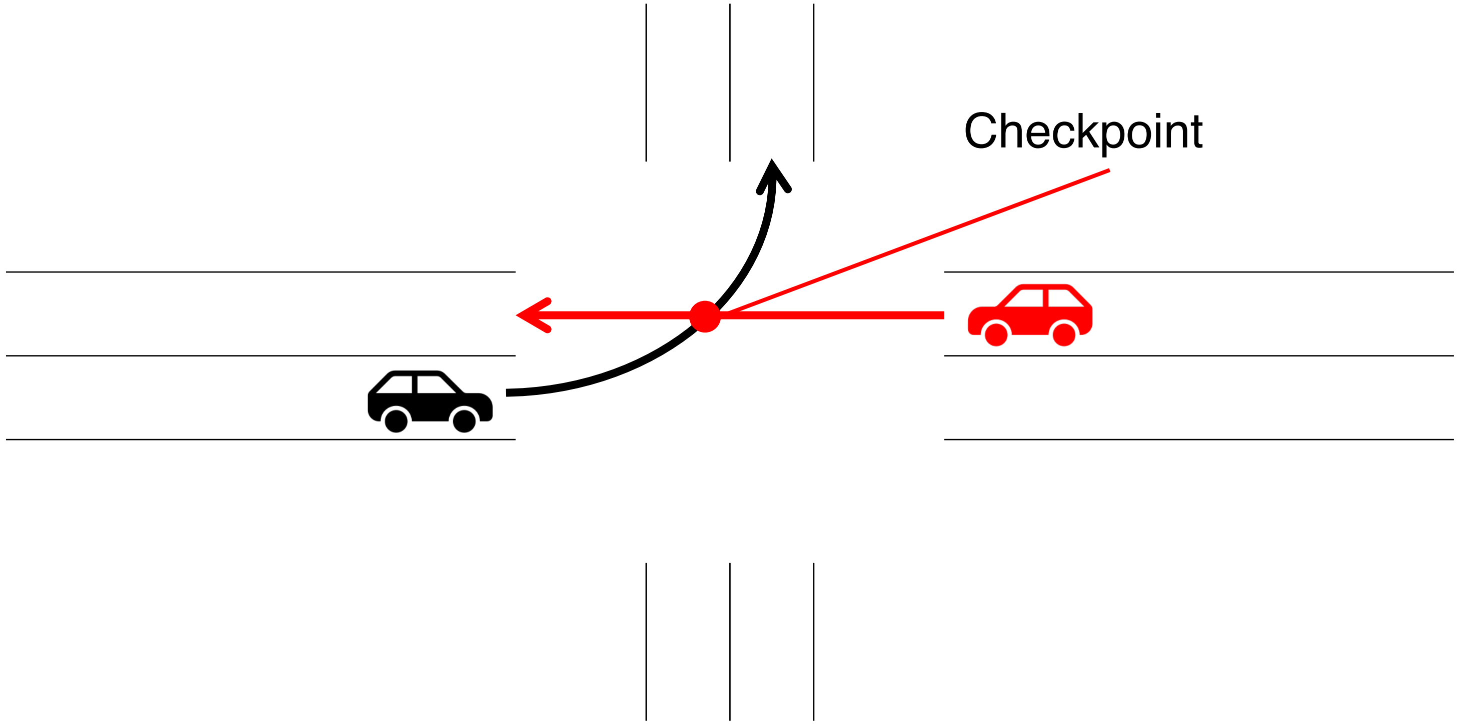
- Lane changing model



Shadow vehicle

CityFlow Design

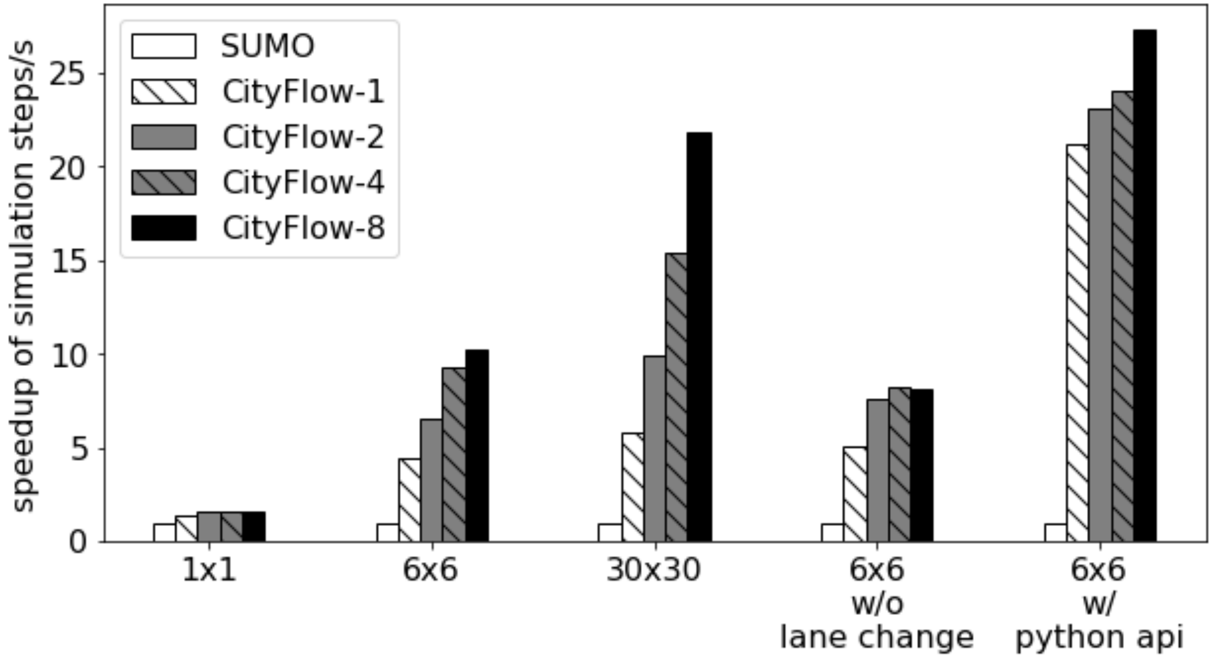
- Intersection model



CityFlow

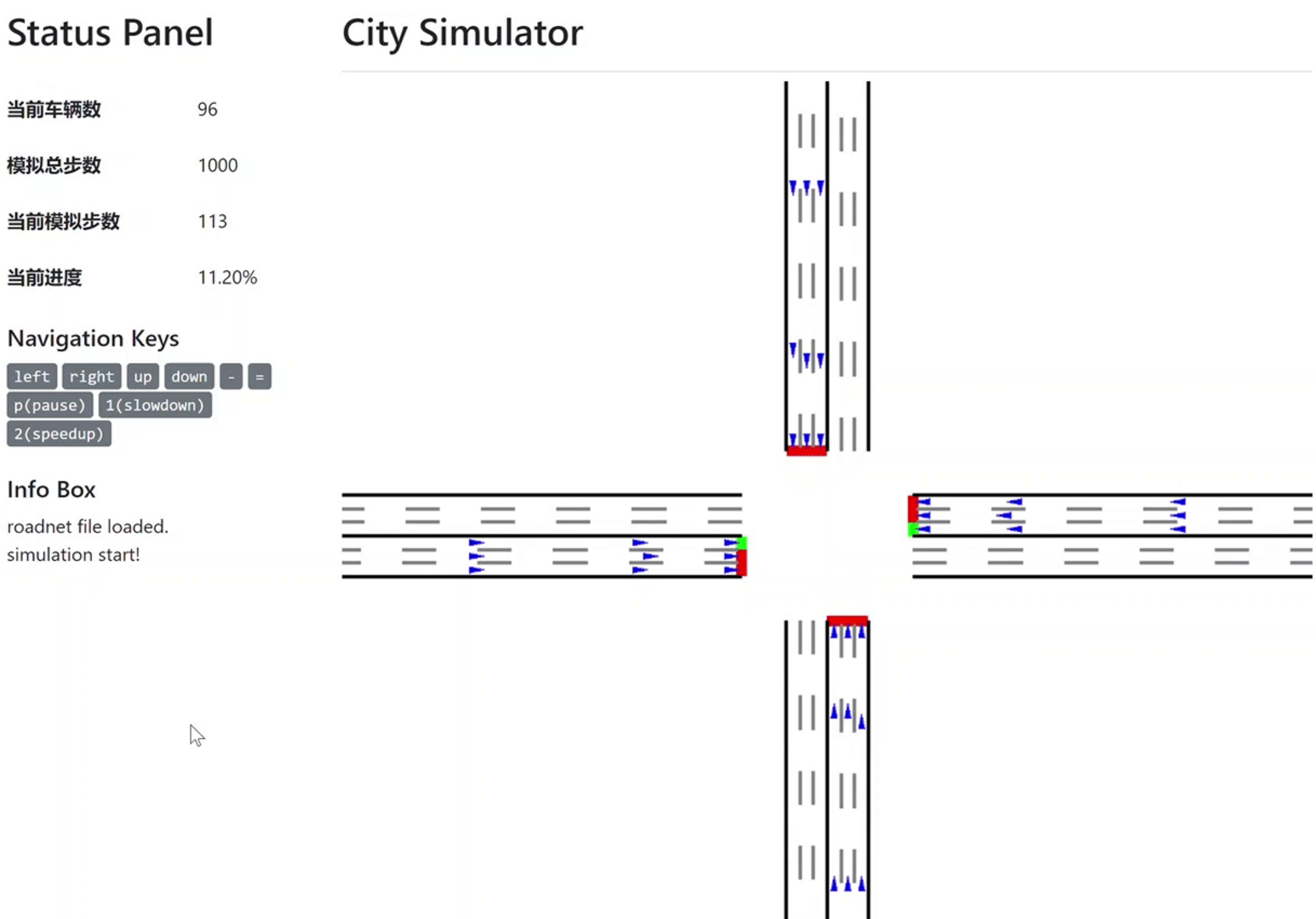
https://github.com/cityflow-project/CityFlow

- Focus on speed
    - Data structure design
    - Simulation algorithm design
    - Multithread
    - Faster python api (compared to SUMO)



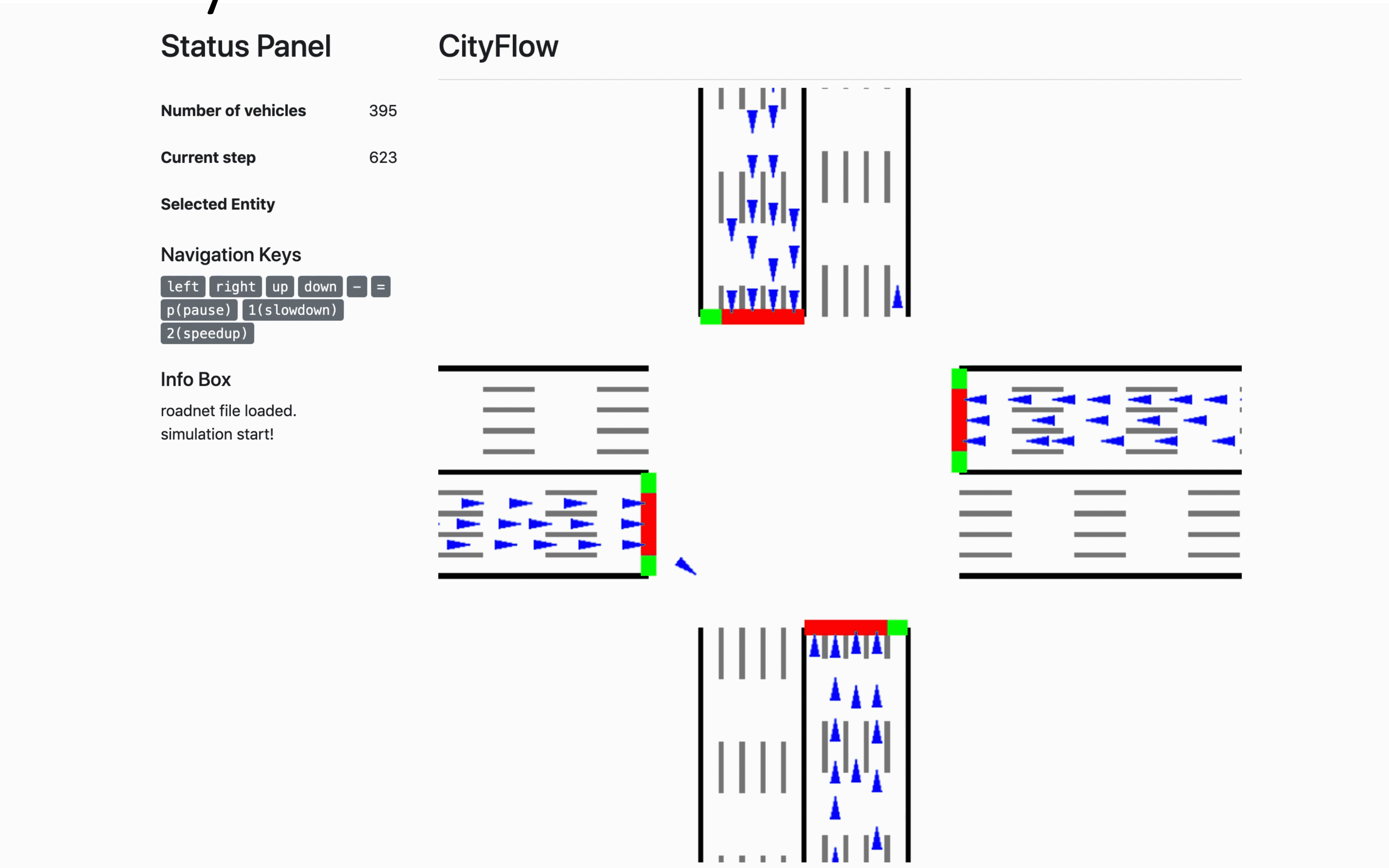
CityFlow

https://github.com/cityflow-project/CityFlow



Huichu Zhang, Siyuan Feng et al. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. WWW 2019 demo

CityFlow https://github.com/cityflow-project/CityFlow



Huichu Zhang, Siyuan Feng et al. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. WWW 2019 demo

CityFlow

https://github.com/cityflow-project/CityFlow

Status Panel

Number of vehicles 33

Current step 13

Selected Entity

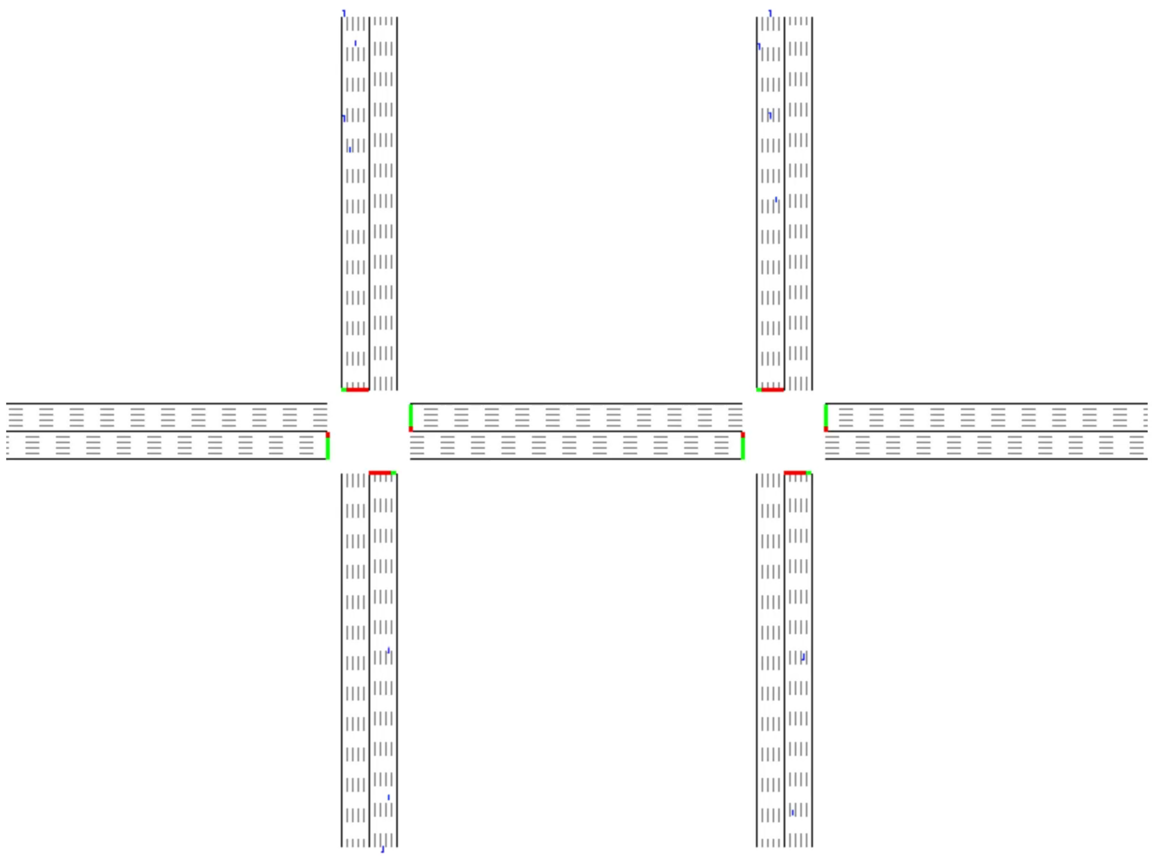
Navigation Keys

`left` `right` `up` `down` `-` `=`
`p(pause)` `1(slowdown)`
`2(speedup)`

Info Box

roadnet file loaded.
simulation start!

CityFlow



Huichu Zhang, Siyuan Feng et al. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. WWW 2019 demo

Compare Good/Bad Traffic Control

当前模拟步数 0

当前进度 0

Navigation Keys

`left` `right` `up` `down` `-` `=`
`p(pause)` `1(slowdown)`
`2(speedup)`

Info Box

Worse control

roadnet file loaded.

当前模拟步数 0

当前进度 0

Navigation Keys

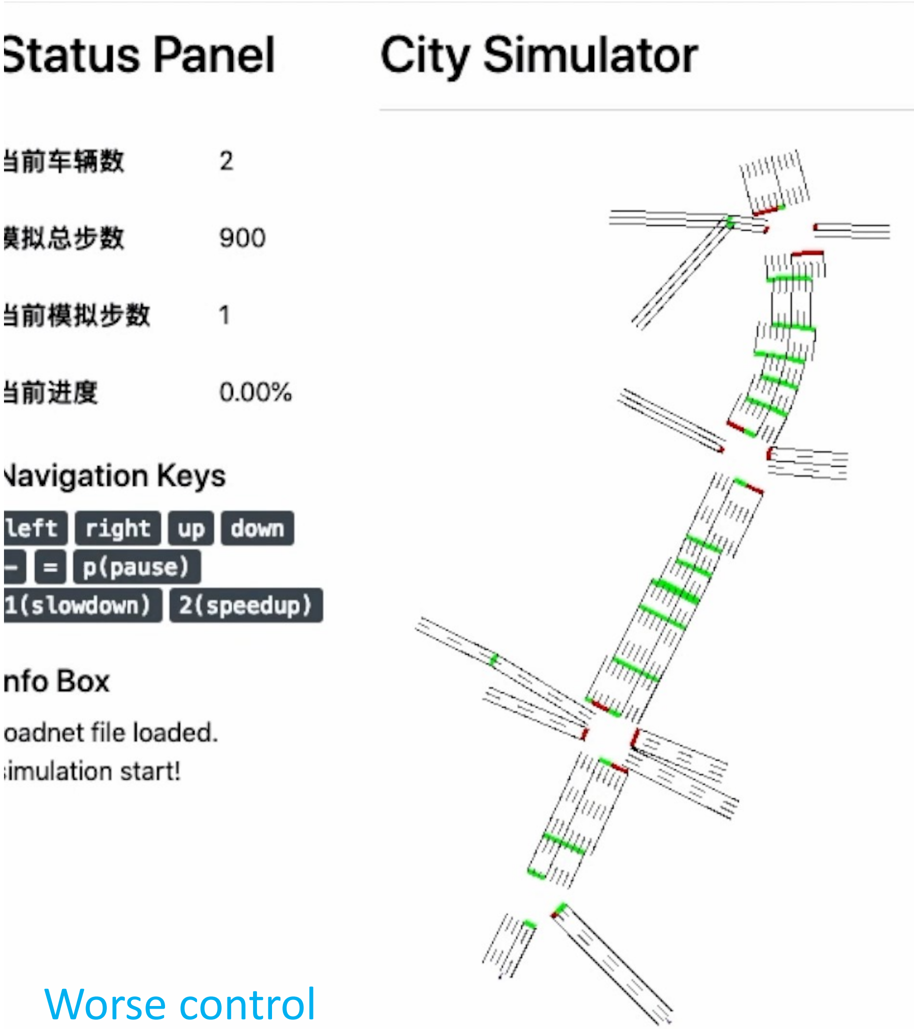
`left` `right` `up` `down` `-` `=`
`p(pause)` `1(slowdown)`
`2(speedup)`

Info Box

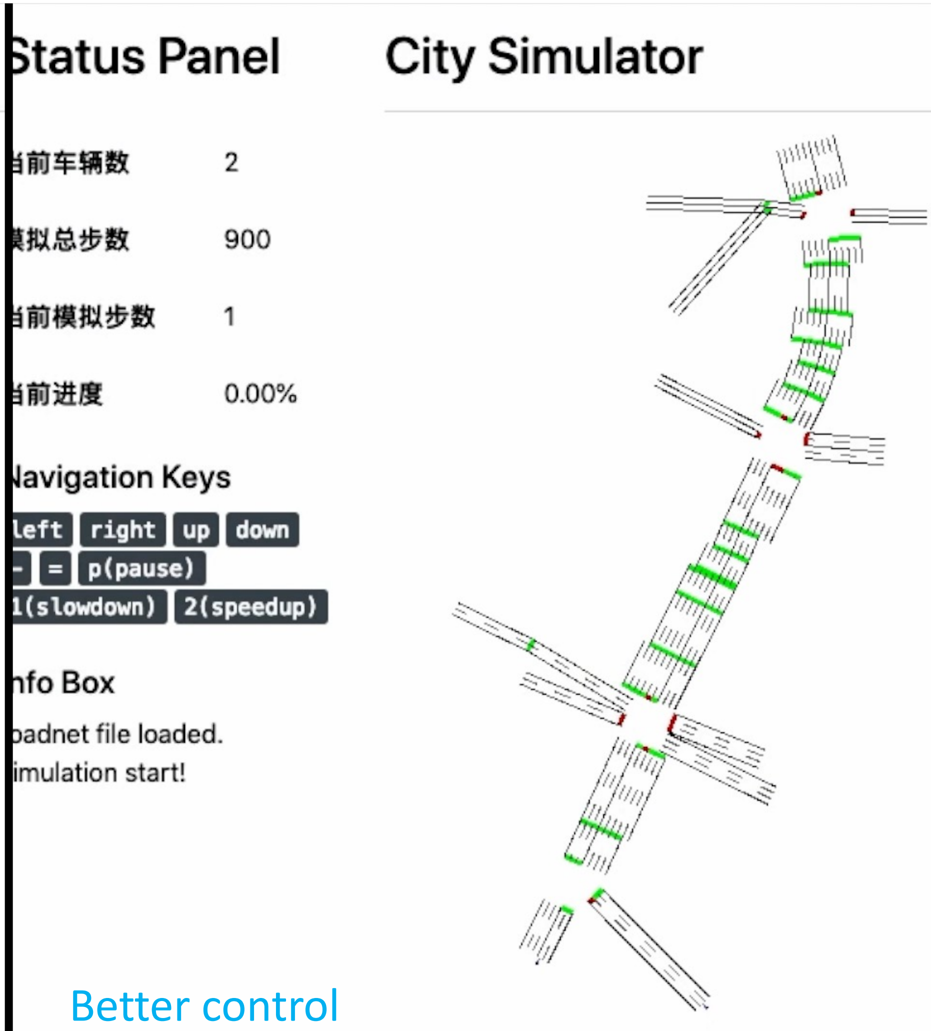
Better control

roadnet file loaded.

Test on Real-World Road Network:
Los Angeles (4 Intersections)



Worse control



Better control

Real World City Simulator: Manhatton
(2510 Intersections)

Real World City Simulator: Manhatton
(2510 Intersections)

Status Panel

当前车辆数	940
模拟总步数	1800
当前模拟步数	294
当前进度	16.28%

Navigation Keys

`left` `right` `up` `down` `-` `=` `p(pause)` `1(slowdown)` `2(speedup)`

Info Box

roadnet file loaded.
simulation start!

City Simulator

TSCC2050

- Use a customized version of CityFlow as backend

智能交通灯调度落地

杭州城市大脑 – 调控效果展示

以上塘-中河高架为例

早晚通勤时长

早高峰 – 北向南（上塘–中河高架）

从25分钟缩减到19分钟

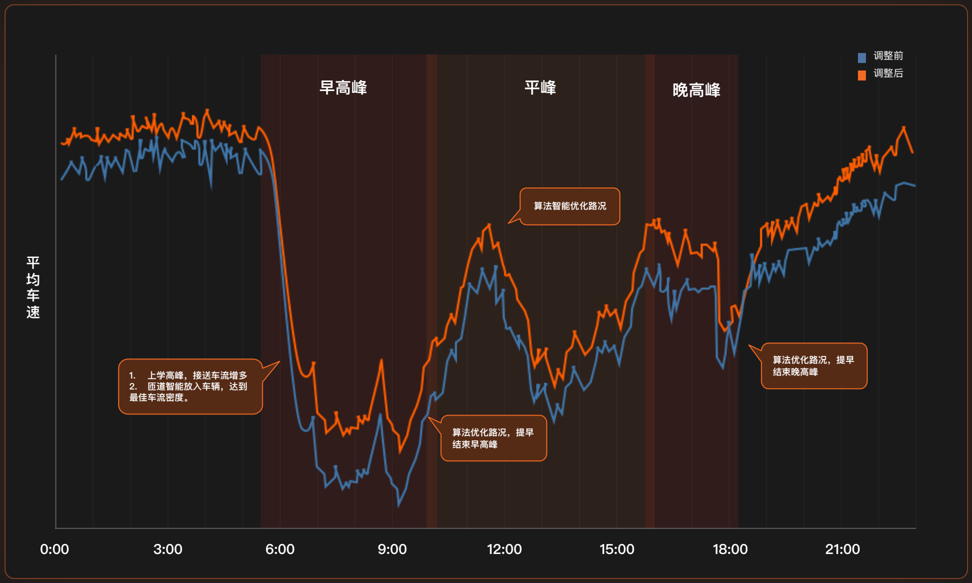
减少24%

晚高峰 – 南向北（中河–上塘高架）

从26.5分钟缩减到22.5分钟

减少15%

全天调控效果



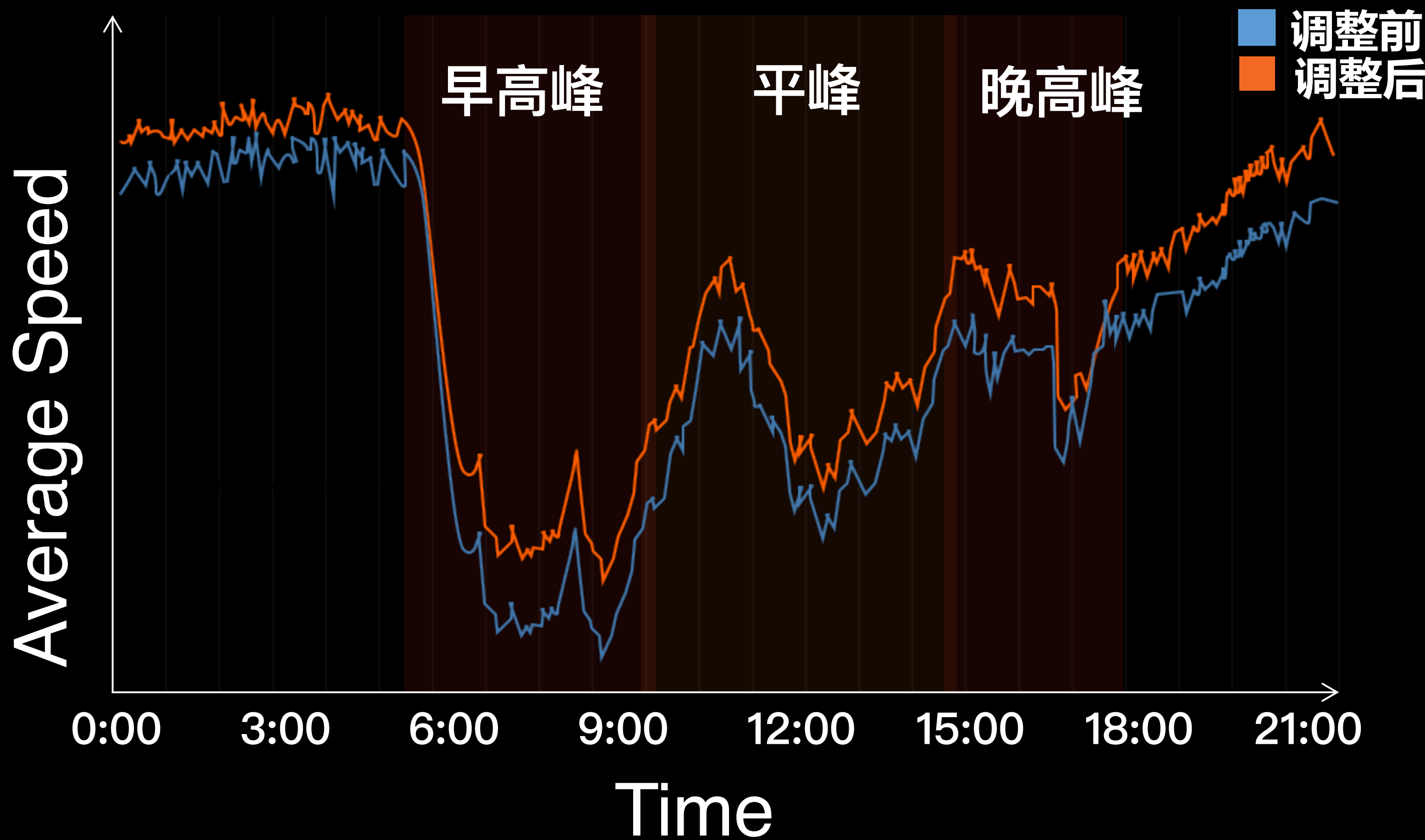
减少排放CO_2共14.5吨

CO_2 14.5吨

=

3000棵树

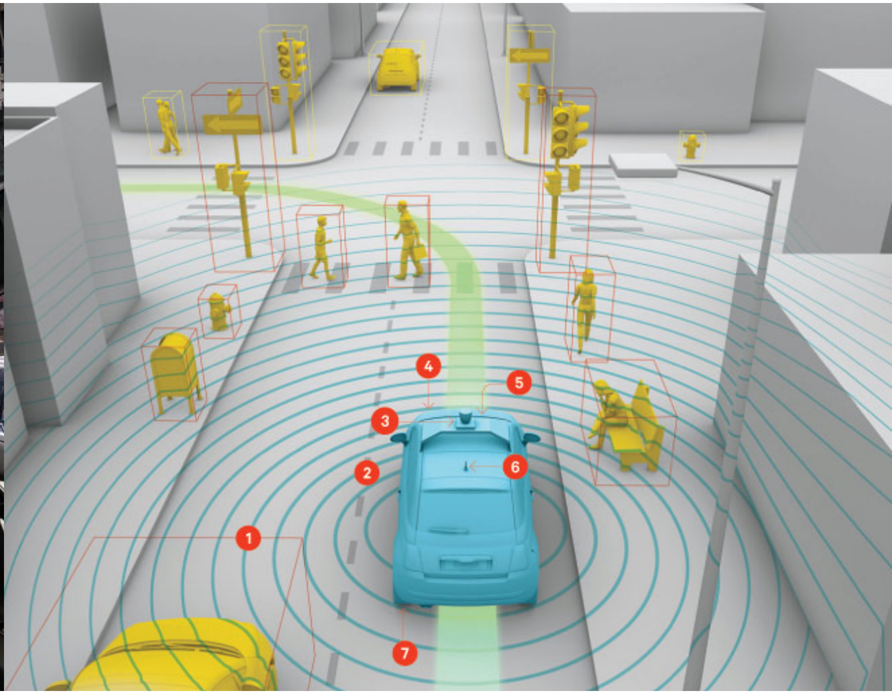
智能交通灯调度落地


早高峰
平峰
晚高峰
调整前
调整后
Average Speed
0:00
3:00
6:00
9:00
12:00
15:00
18:00
21:00
Time

Featured Simulators

- Discrete world: MAgent
    - https://github.com/geek-ai/MAgent
- Continuous world: Cityflow
    - https://github.com/cityflow-project/CityFlow/
- Self-driving cars: SMARTS
    - https://github.com/huawei-noah/SMARTS

Autonomous Driving



Step 1: Perception

Recognize the objects around & build the 3D world

Step 2: Control

Make action decisions in the built world

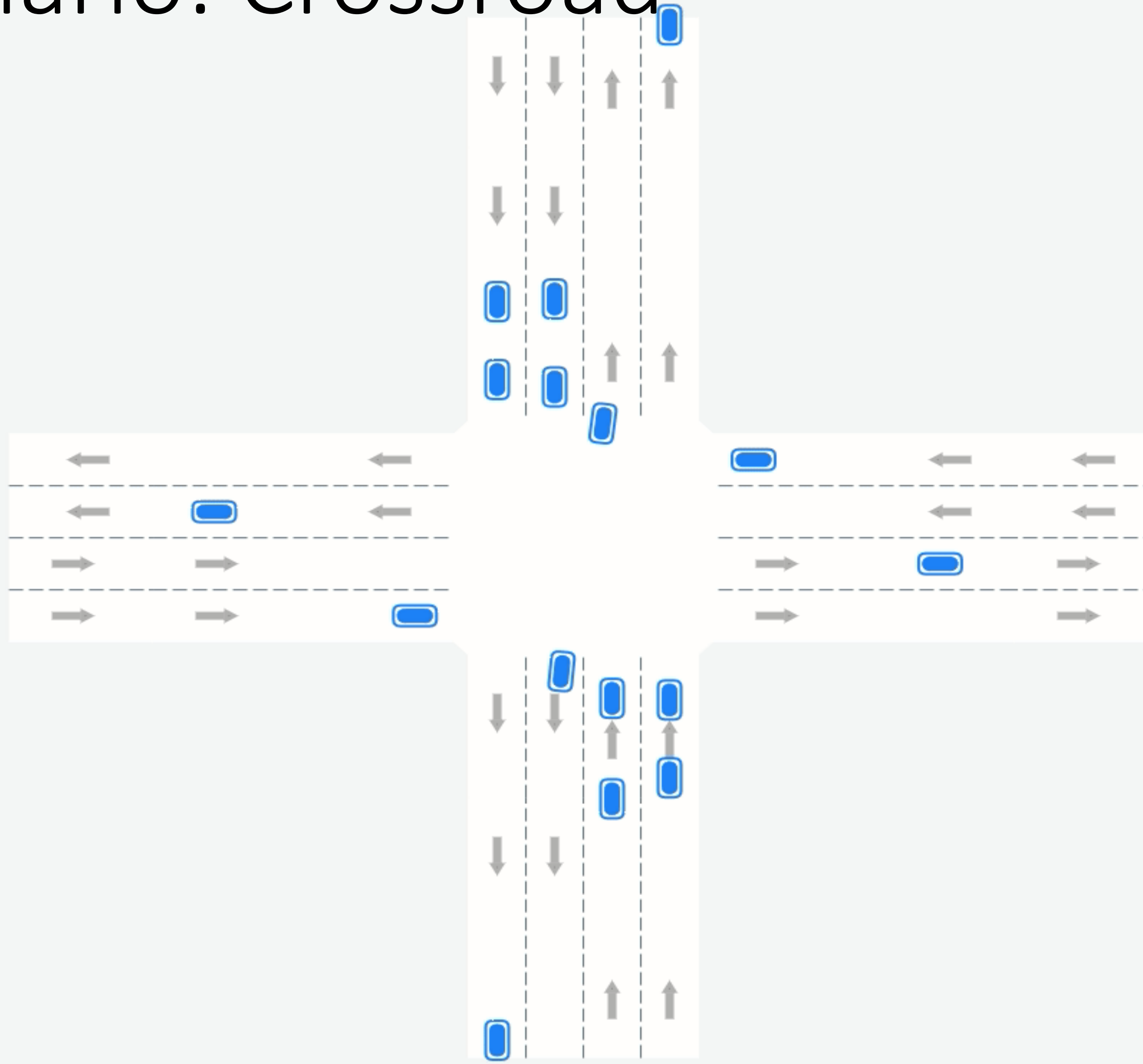
SMARTS (with Huawei Team) https://github.com/huawei-noah/SMARTS

- A scalable multi-agent learning simulator for autonomous driving control
- Flexibility and high efficiency in the physical simulation and interacting with multi-agent reinforcement learning algorithms

Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Scenario: Multi-lane Cruising

Scenario: Crossroad



Scenario: On-ramp

Multi-Agent Learning for Auto-Driving

Level	Description	Possible MARL Approach to Double Merge
M0	Rule-based planning & control without adaptive learning	N/A
M1	Single-agent learning with ignorance of other learning agents	Learning agent could learn to implicitly anticipate how other agents will react to its own actions but will likely suffer from non-stationarity and lack generalizability [2].
M2	Decentralized multi-agent learning with opponent modeling	MARL to model other agents, e.g. "high likely they will yield to me if I start changing to their lane given how they have been driving in the past few seconds." [3] [4]
M3	Coordinated multi-agent learning and independent execution	Coordinated learning of what to expect even when there is no explicit coordination during execution: "some of them will give me the gap." [5]
M4	Local (Nash) equilibrium oriented multi-agent learning	Learn as a group to achieve a certain equilibrium such that each will get the chance to go through the intersection without too much trouble. [6, 7]
M5	Social welfare oriented multi-agent learning	Learn broader repercussions of our actions as a group, e.g. "if I force to the left now, I will cause a congestion on the left road, because of the fast and heavy traffic there."

Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

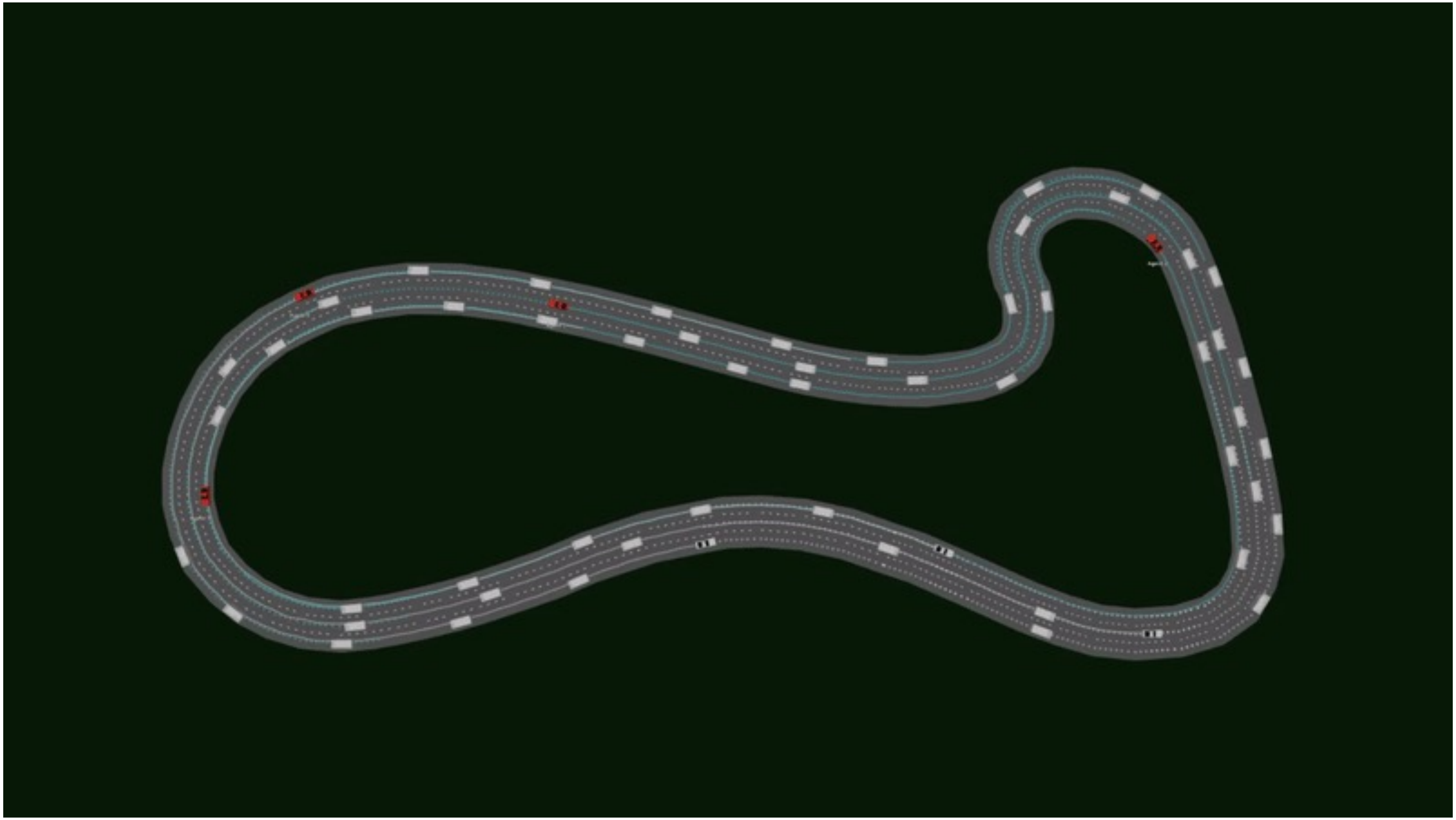
SMARTS:
Scalable Multi-Agent RL Training School

- SMARTS is an open-source scalable multi-agent reinforcement learning platform for autonomous driving.

Distributed/Parallel Computing	High fidelity	Interaction Scenario Design	User friendly rendering

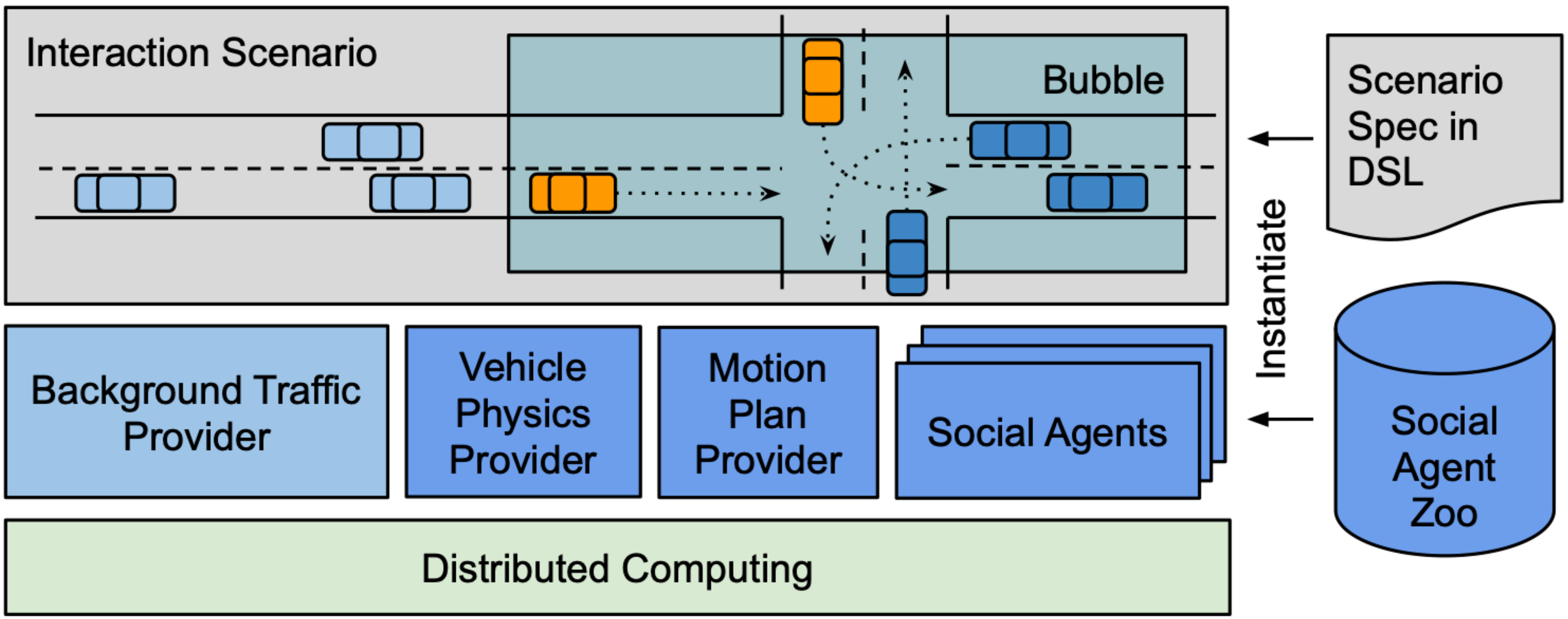
Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Overview Demo of SMARTS



https://github.com/huawei-noah/SMARTS

SMARTS Architecture

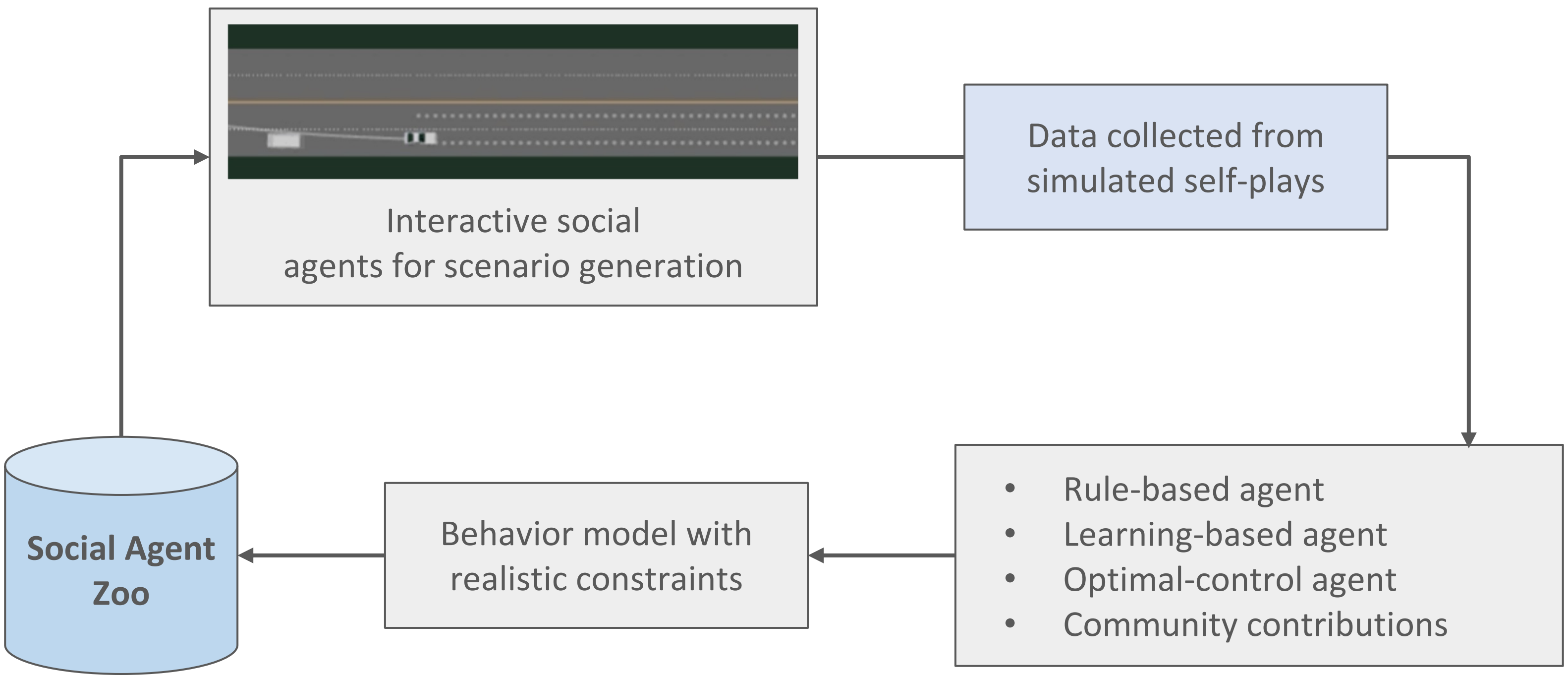


Interaction scenarios are instantiated based on a (domain specific language) DSL specification. Social agents are instantiated from the Social Agent Zoo. Orange vehicles are controlled by learning agents, dark blue vehicles by social agents, light blue ones by traffic provider. All providers and agents in principle run in their own processes, possibly remotely.

Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Bootstrapping Realistic Interaction

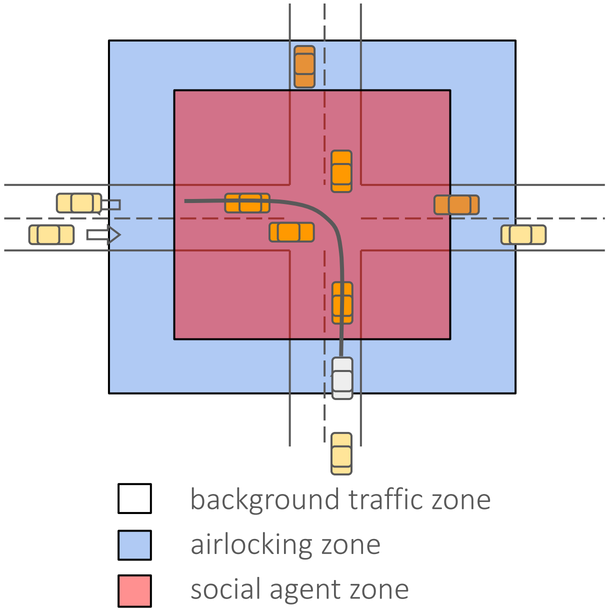
- Key contextual factors of realistic and diverse:
  - Physics
  - Road users' behavior
  - Road structure & regulations

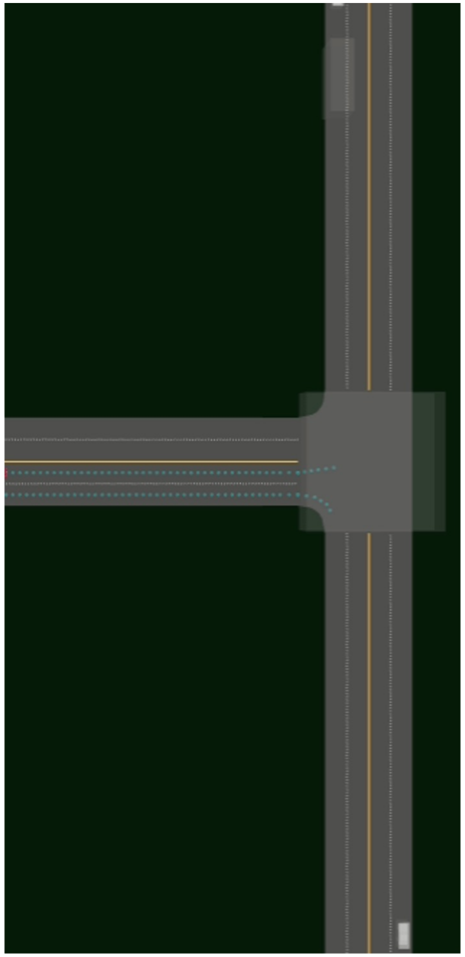


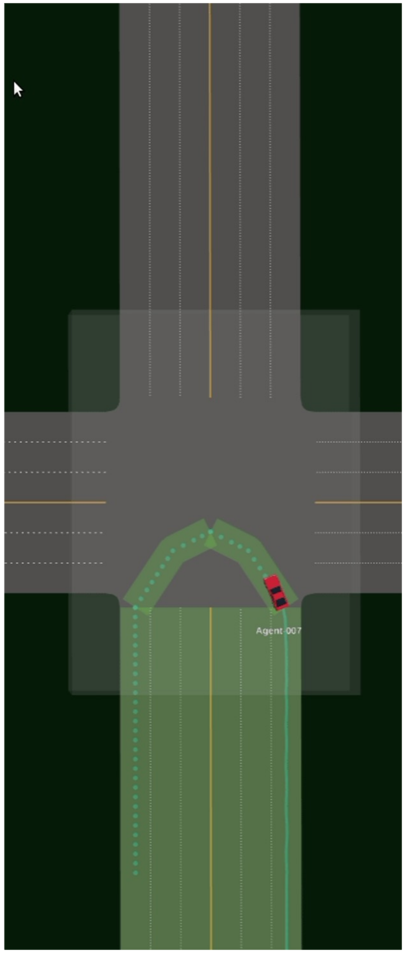
Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Bubble Mechanism

- The bubble mechanism allows SMARTS to scale without sacrificing interaction relism.
- Goal: support large-scale simulation and distributed computing.



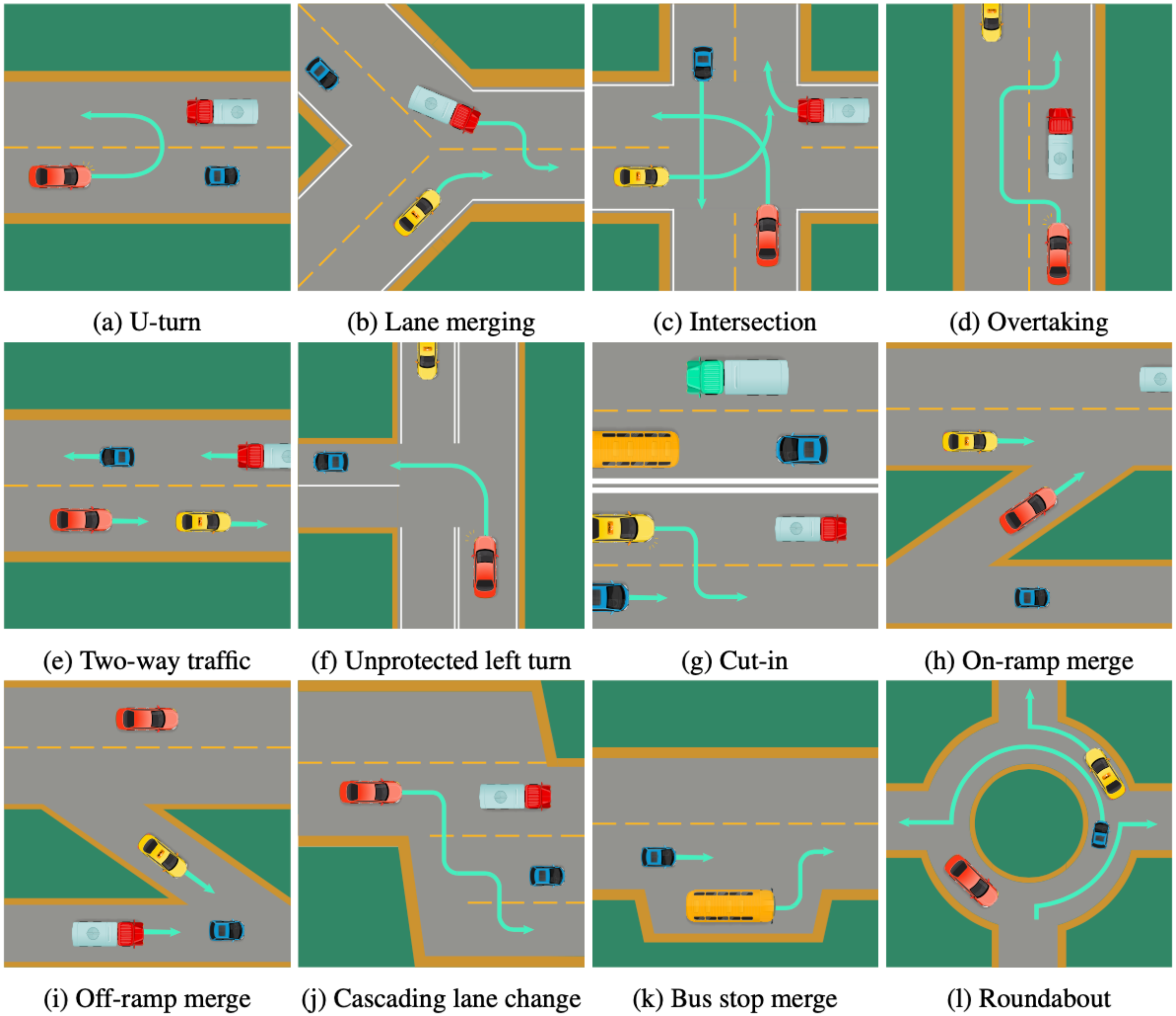




Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Interaction Scenario Library

- Using the scenario DSL of SMARTS, we can create numerous scenarios that vary in road structure and traffic.
- Scenarios provide rich traffic flows and road conditions to help us study behavior and driving strategies.



(a) U-turn (b) Lane merging (c) Intersection (d) Overtaking (e) Two-way traffic (f) Unprotected left turn (g) Cut-in (h) On-ramp merge (i) Off-ramp merge (j) Cascading lane change (k) Bus stop merge (l) Roundabout

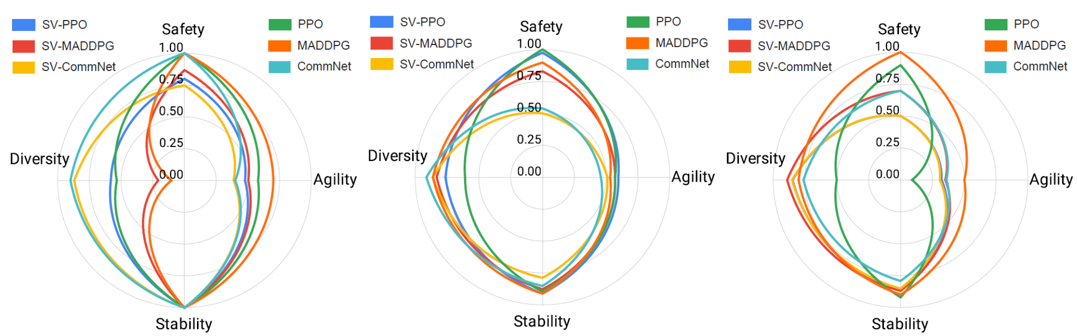
Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

MARL Benchmarking

- Distributed training supported.
- Rich metrics.
- Customizable observation, action (controller) and reward function.
- Algorithms/Baselines
  - Independent learning
  - Centralized/decentralized learning
  - Networked agent learning
  - …

Evaluation metrics of multi-agent AD

Metric	Type	Description
Collision Rate	Performance	Collision rate over a group of episodes.
Completion Rate	Performance	Mission completion over a group of episodes.
Generalization	Performance	Robustness of algorithms to scenario variation.
Safety	Behavior	Integrated metrics, e.g. non-collision rate.
Agility	Behavior	Integrated metrics, e.g. speed.
Stability	Behavior	Integrated metrics for driving smoothness.
Control Diversity	Behavior	Preference for longitudinal or lateral control.
Cut-in Ratio	Behavior	Probability of cut-in in traffic flow.
Stochasticity	Behavior	Stochasticity of decision making.
Collaborative	Game theory	Compatible interests, e.g. ratio of giving way.
Competitive	Game theory	Conflicting interests, e.g. ratio of overtaking.



(a) Two-Way (b) Double Merge (c) Intersection

Results on behavior metrics.
"SV-" represents the algorithms interacting with social vehicles.

Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

SMARTS Supported MARL Algorithms

Paradigm	Algorithm	Communication	Framework
Fully centralized training	BiCNet ⋆	Yes	malib
	CommNet ⋆	Yes	malib
Fully decentralized	Indepedent Q	No	RLlib
	Independent PG	No	RLlib
	Independent AC	No	RLlib/malib
	PR2	No	malib
	ROMMEO	No	malib
	Supervised Opponent Modeling	No	malib
CTDE	Centralized V	No	RLlib
	MAAC ⋆	No	RLib
	MADDPG	No	malib
	MF-AC/Q ⋆	No	malib
	COMA	No	PyMARL
	VDN	No	PyMARL
	QMIX	No	PyMARL
	QTRAN	No	PyMARL
	MAVEN	No	PyMARL
	Q-DPP ⋆	No	PyMARL
Networked agent learning	Networked Fitted-Q ⋆	graph	RLlib

Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Overall Evaluation Results

Average collision rate / completion rate of selected baselines

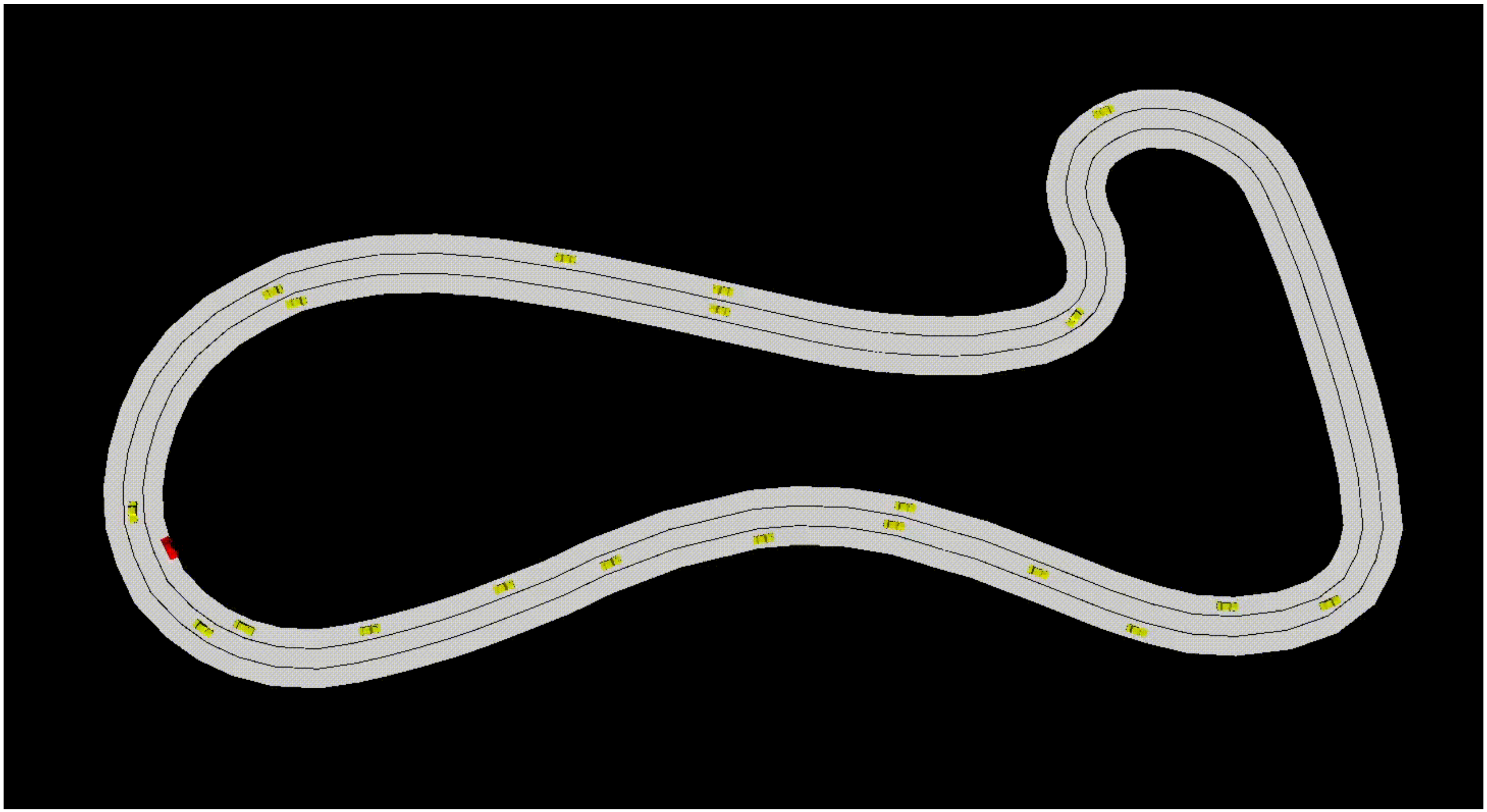
Algorithm	Scenario - No Social Vehicle			Scenario - Random Social Vehicle		
	Two-Way	Double Merge	Intersection	Two-Way	Double Merge	Intersection
DQN	0/0.97	0.77/0.23	0.83/0.20	0.40/0.60	0.60/0.23	0.92/0.05
PPO	**0/1**	**0/1**	0.1/0.07	0.25/0.75	**0.02/0.98**	0.50/0.45
MAAC	**0/1**	0.42/0.58	**0/1**	0.25/0.75	0.42/0.6	0.32/0.68
MFAC	0/0.8	0.6/0.4	0.54/0.4	0.45/0.5	0.7/0.3	0.62/0.37
Net-Q	0/0.3	0.7/0.25	0.4/0.23	0.4/0.2	0.8/0.2	0.75/0.2
CommNet	0/0.96	0.46/0.45	0.3/0.7	0.25/0.65	0.5/0.5	0.5/0.45
MADDPG	**0/1**	0.1/0.9	**0/1**	**0.13/0.87**	0.17/0.8	**0.30/0.7**

Observations from above table

- PPO is the empirically best single-agent RL method
- MADDPG is the empirically best multi-agent RL method

Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

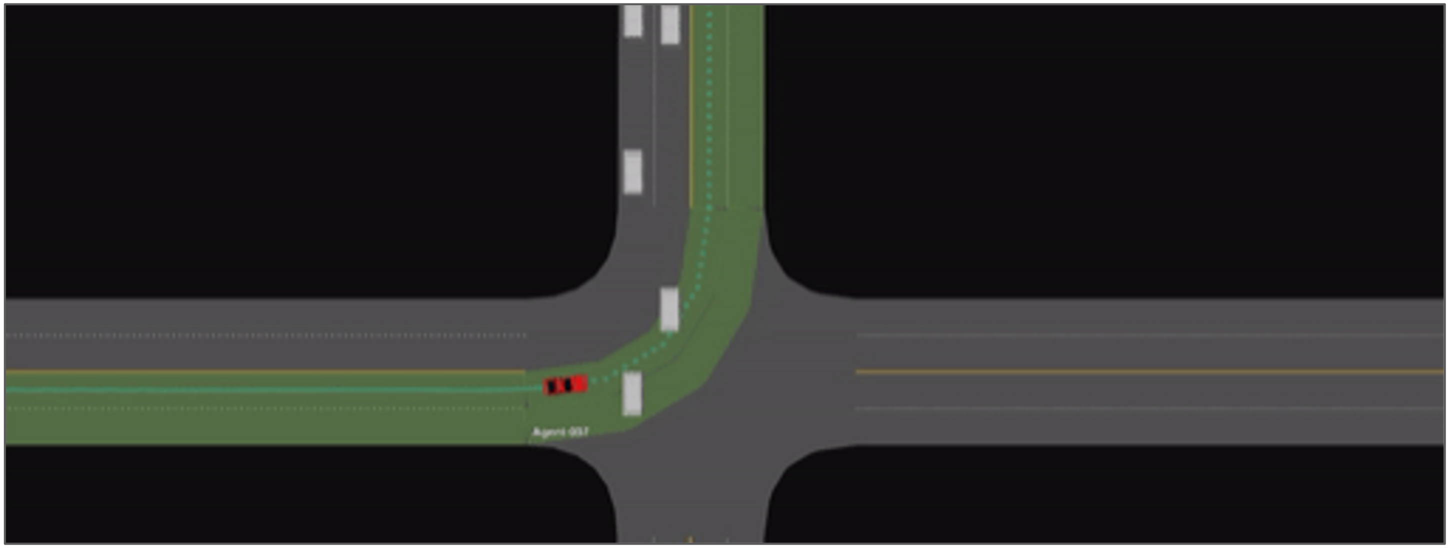
Demo: Single Agent with PPO



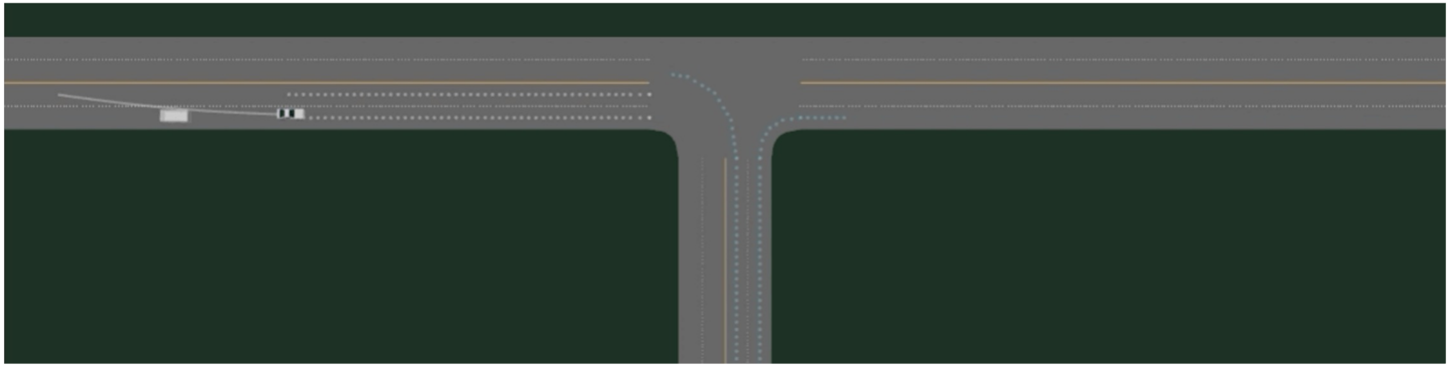
Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Demo: Intersection (Unprotected Left Turn)

Hard flow



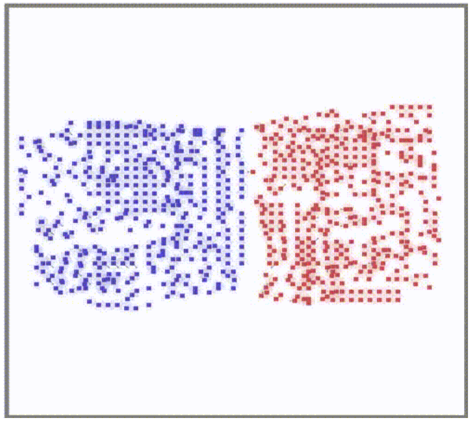
Easy flow



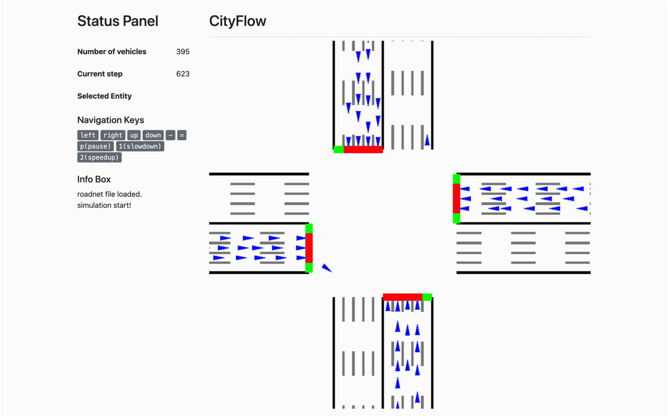
Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.

Summary of Many-Agent RL

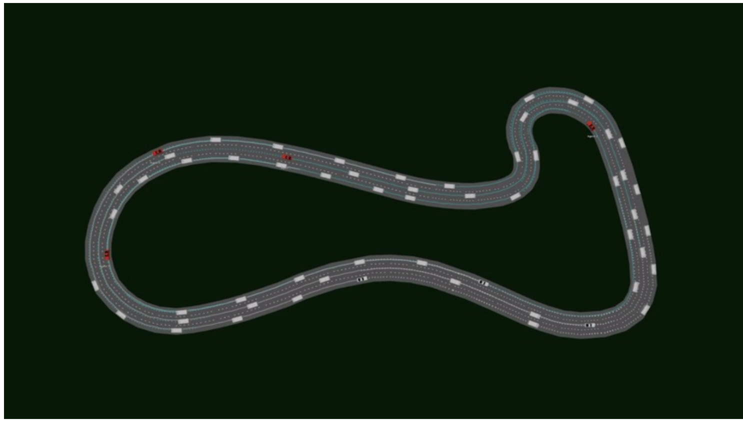
- Main difficulties for many-agent RL
  - Computational complexity
  - Complicated agent interactions
  - Highly dynamic neighborhood
- Possible solutions to many-agent RL
  - Mean field approximation and factorized value functions
  - Scalable MARL platforms



MAgent



CityFlow

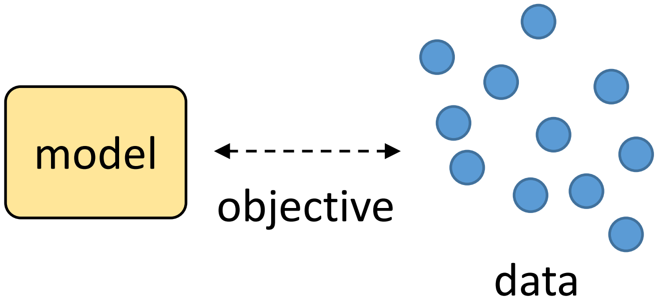


SMARTS

Summary from Machine Learning Perspective

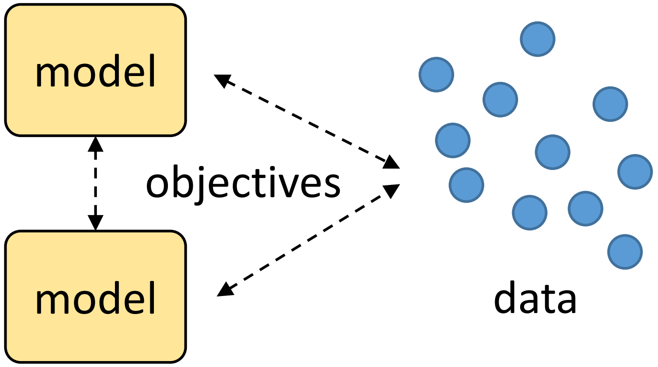
- Traditional machine learning is to build
    - a loss function
    - a likelihood estimation
    - an expectation of value

  from a machine and the training data and to optimize the objective



- Multi-agent machine learning is to build
    - a loss function
    - a likelihood estimation
    - an expectation of value

  from the two machines and the training data and to optimize the objective



Summary of MARL

- Machine learning paradigm shift
  - From prediction to decision making (RL)
  - From single-agent to multi-agent to many-agent
  - An intersection with game theory is essential
- Multi-agent RL
  - Centralized training (with decentralized execution)
  - Decentralized training (with networked agents)
- Many-agent RL
  - Challenges
    - Computational complexity
    - Complicated agent interactions
    - Highly dynamic neighborhood
  - Possible solutions
    - Mean field approximation and factorized value functions
    - Scalable MARL platforms: Magent and CityFlow etc.

Thank You!
Questions?

Weinan Zhang

Associate Professor at Shanghai Jiao Tong University

http://wnzhang.net

Related references

1. Yaodong Yang et al. Mean Field Multi-Agent Reinforcement Learning. ICML 2018.
2. Ming Zhou et al. Factorized Q-Learning for Large-Scale Multi-Agent Systems. DAI 2019.
3. Lianmin Zheng et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. AAAI 2018.
4. Huichu Zhang et al. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. WWW 2019.
5. Ming Zhou et al. SMARTS: A Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. CoRL 2020.
6. Muning Wen et al. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. NeurIPS 2022.
7. Xihuai Wang et al. Order Matters: Agent-by-agent Policy Optimization. ICLR 2023.