Cooperative Multi-Agent Reinforcement Learning (Coop MARL)

Part 1. Introduction and Value-based Methods

Acknowledgement: Most slides were contributed by 廖唯辰、何國豪, and organized by 廖唯辰.

Outline

- Introduction to Cooperative MARL
- Value-based Cooperative MARL
- Policy-based Cooperative MARL

Key Points Today

- Introduction to Cooperative Multi-Agent RL
- Value-based methods
  - CTDE framework
  - Algorithms
    - VDN
    - QMIX

Reference

- [marlgt]
  - Yang, Yaodong, and Jun Wang. "An overview of multi-agent reinforcement learning from game theoretical perspective." arXiv preprint arXiv:2011.00583 (2020). (https://arxiv.org/abs/2011.00583)
- [RL]
  - Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [gcoop]
  - Yang, Y. (n.d.). A GENERAL SOLUTION FRAMEWORK FOR COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING. Retrieved April 16, 2023, from https://www.yangyaodong.com/_files/ugd/ddd18b_969600e4558b43f29a250e573d618cbf.pdf
- [mc]
  - Cooperative MARL slides by Kuo-Hao Ho

Reference

- [vdn]
  - Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." arXiv preprint arXiv:1706.05296 (2017). (https://arxiv.org/abs/1706.05296)
- [qmix]
  - Rashid, Tabish, et al. "Monotonic value function factorisation for deep multi-agent reinforcement learning." The Journal of Machine Learning Research 21.1 (2020): 7234-7284. (https://arxiv.org/abs/1803.11485)
- [qtran]
  - Son, Kyunghwan, et al. "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning." International conference on machine learning. PMLR, 2019. (https://proceedings.mlr.press/v97/son19a.html)

Reference

- [maddpg]
  - Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in neural information processing systems 30 (2017). (https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html)
- [coma]
  - Foerster, Jakob, et al. "Counterfactual multi-agent policy gradients." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018. (https://ojs.aaai.org/index.php/AAAI/article/view/11794)
- [mappo]
  - Yu, Chao, et al. "The surprising effectiveness of ppo in cooperative multi-agent games." Advances in Neural Information Processing Systems 35 (2022): 24611-24624. (https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html)

Reference

- [hatrpo]
  - Kuba, J. G., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J., & Yang, Y. (2022). Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. International Conference on Learning Representations. (https://openreview.net/forum?id=EcGGFkNTxdJ)
- [mat]
  - Wen, Muning, et al. "Multi-agent reinforcement learning is a sequence modeling problem." Advances in Neural Information Processing Systems 35 (2022): 16509-16521. (https://proceedings.neurips.cc/paper_files/paper/2022/hash/69413f87e5a34897cd010ca698097d0a-Abstract-Conference.html)

Introduction to MARL

- Introduction to MARL
- Challenges of MARL
- Cooperative MARL

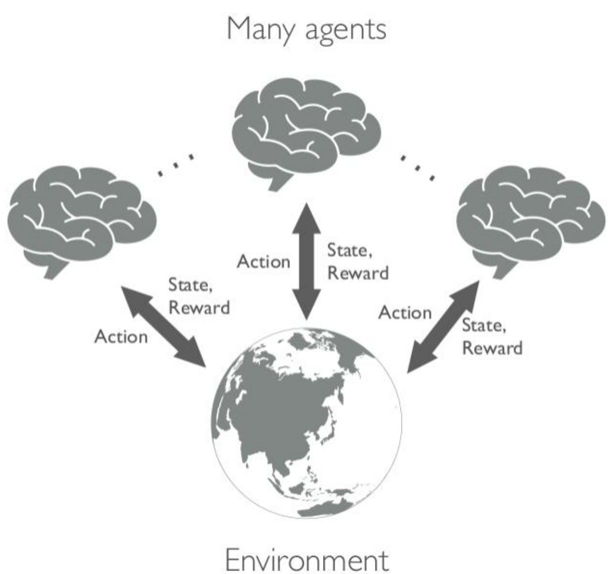
Introduction to MARL

- Introduction to MARL
- Challenges of MARL
- Cooperative MARL

Multi-Agent RL (MARL)

- At least 1 agent(s) in the environment
  - Only 1 agent: degrade to Single-Agent RL

Single-Agent

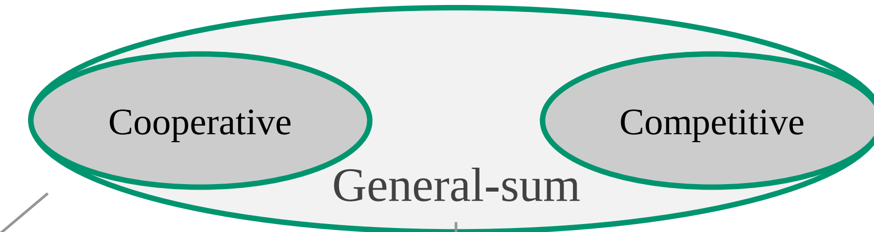


Multi-Agent

Source of image: [marlgt]

Categories of MARL

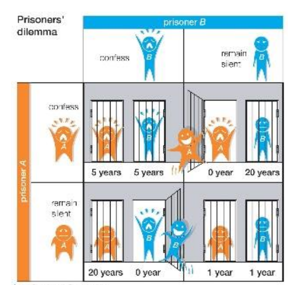
Categorized by Game Types



All agents have identical interests

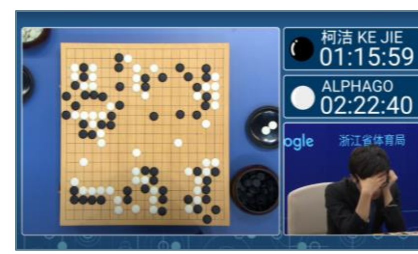
E.g. Robotic Cooperation

Mixture of the other two game types



E.g. Prisoners' Dilemma

Agents share opposite interests and act competitively



E.g. Go

I-Chen Wu

Source of images:
https://www.youtube.com/watch?v=5kIQ0F8iN8U
https://spectrum.ieee.org/mobile-robots-cooperate-to-3d-print-structures
https://www.britannica.com/science/game-theory/N-person-games

Introduction to MARL

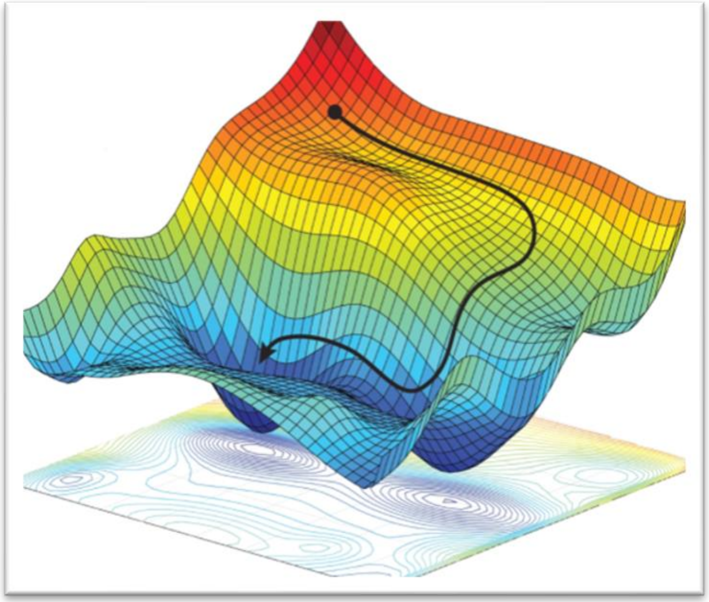
- Introduction to MARL
- Challenges of MARL
- Cooperative MARL

Challenges of MARL

1. Non-stationarity
2. Complexity

Non-Stationarity

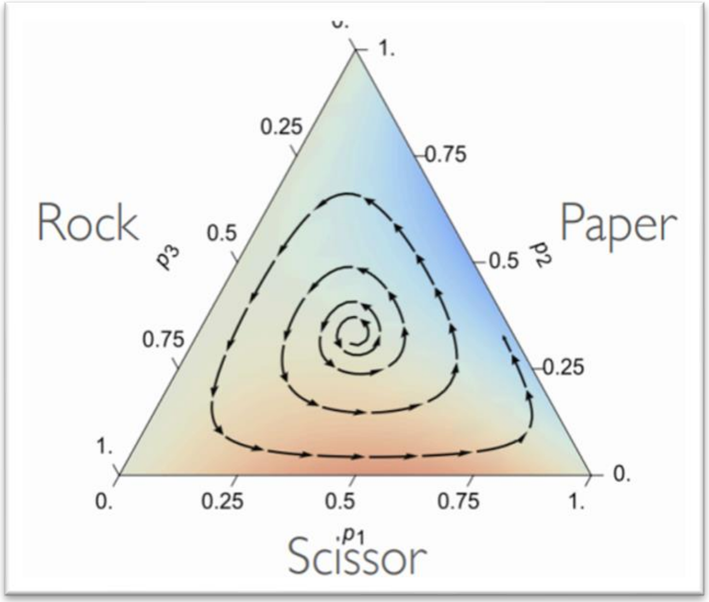
Single-Agent RL



Fixed Loss Landscape

(stationary)

Multi-Agent RL



Dynamic Loss Landscape

(non-stationary)

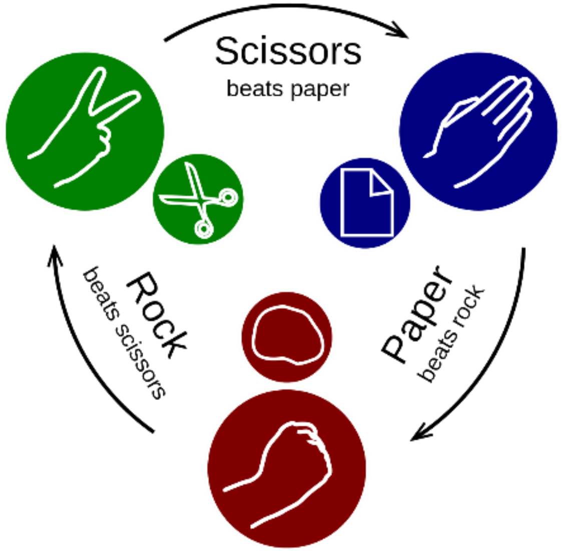
Since all agents are learning!

I-Chen Wu

Source of image: [gcoop]

Non-Stationarity

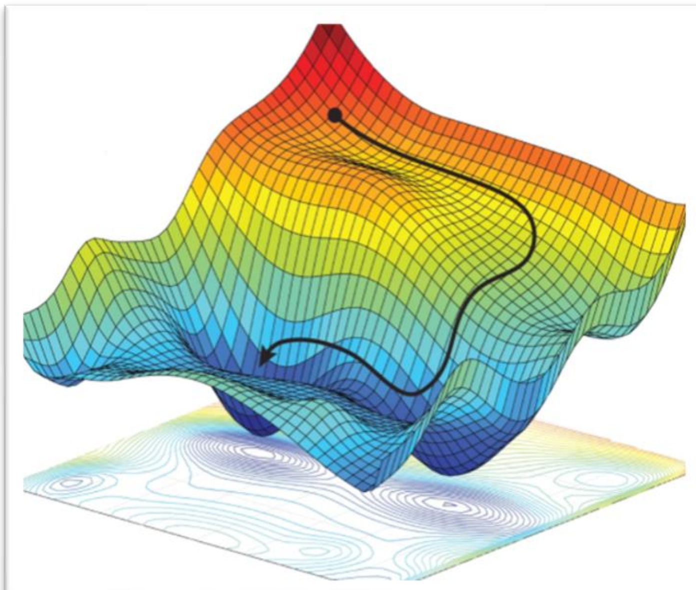
- Example: Rock-Paper-Scissors Game

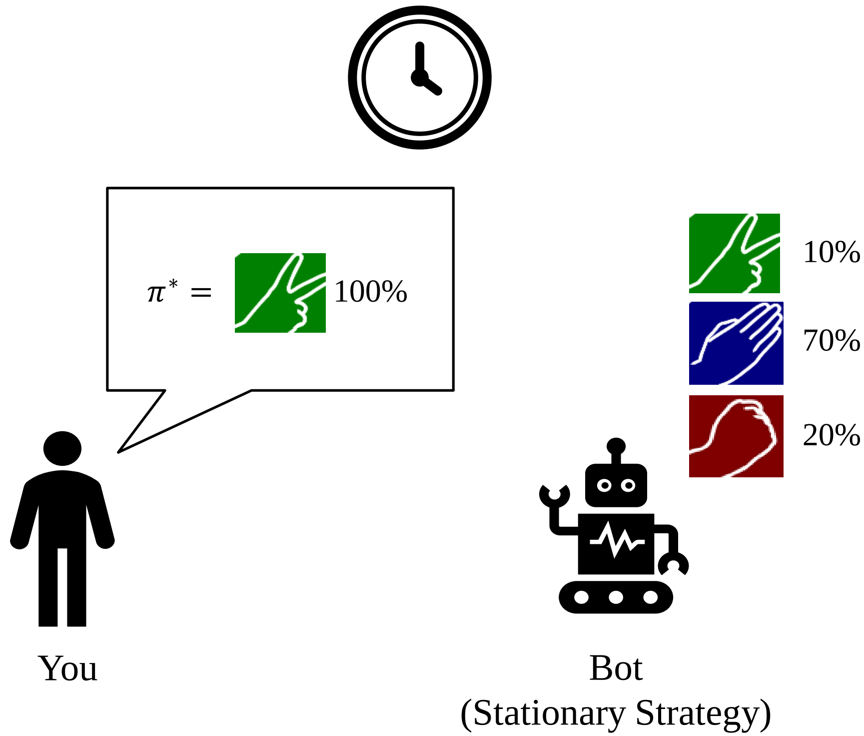


Source of images: https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Single-Agent RL

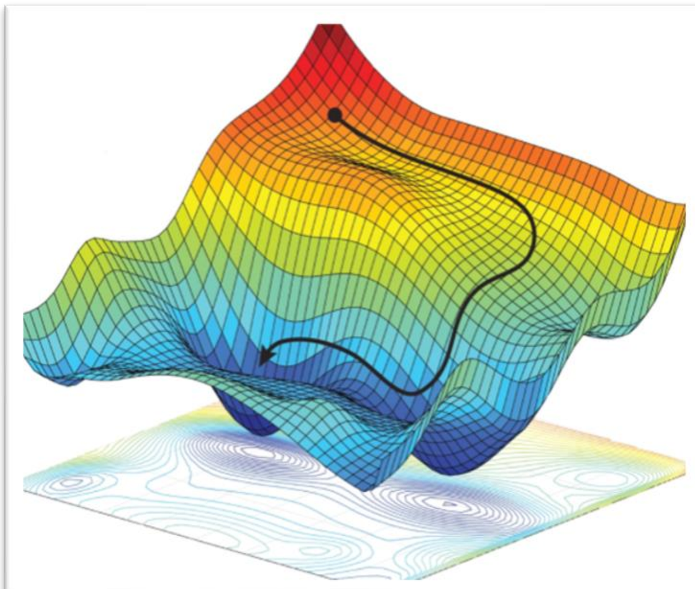


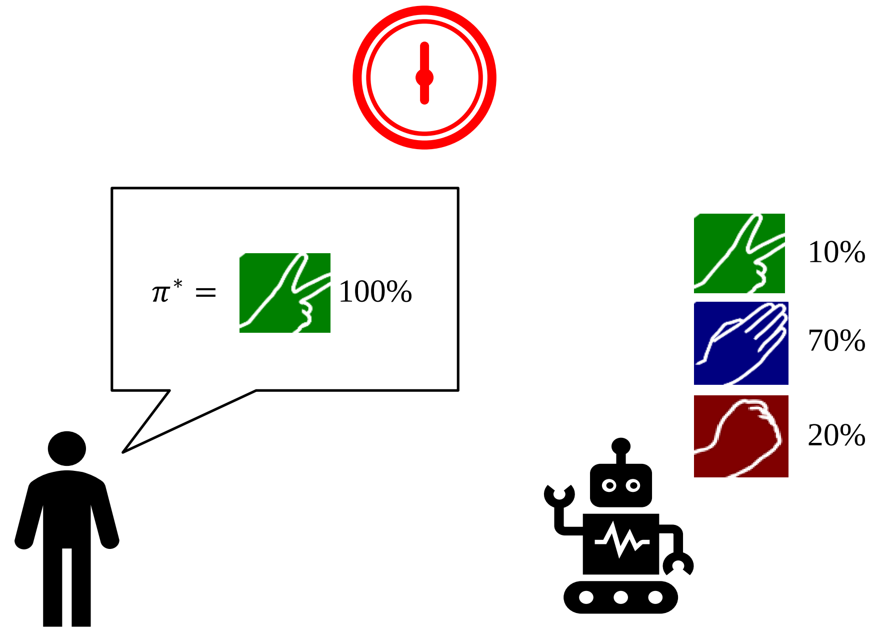


Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Single-Agent RL





You

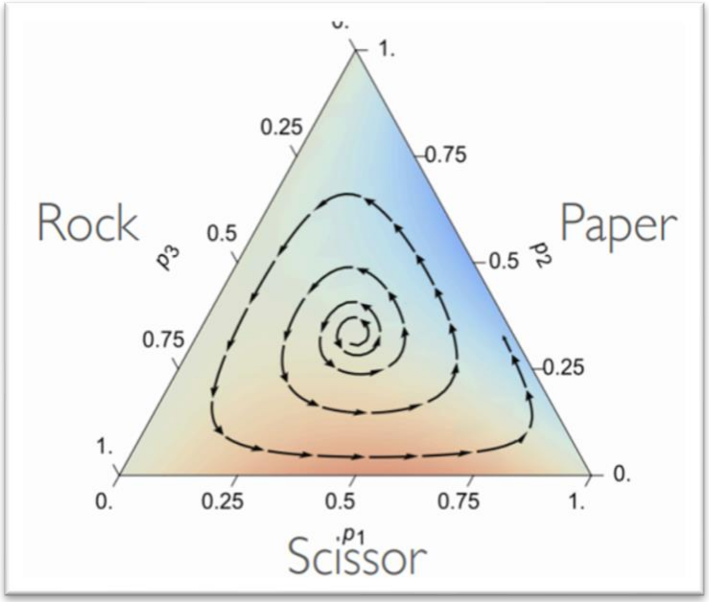
Bot
(Stationary Strategy)

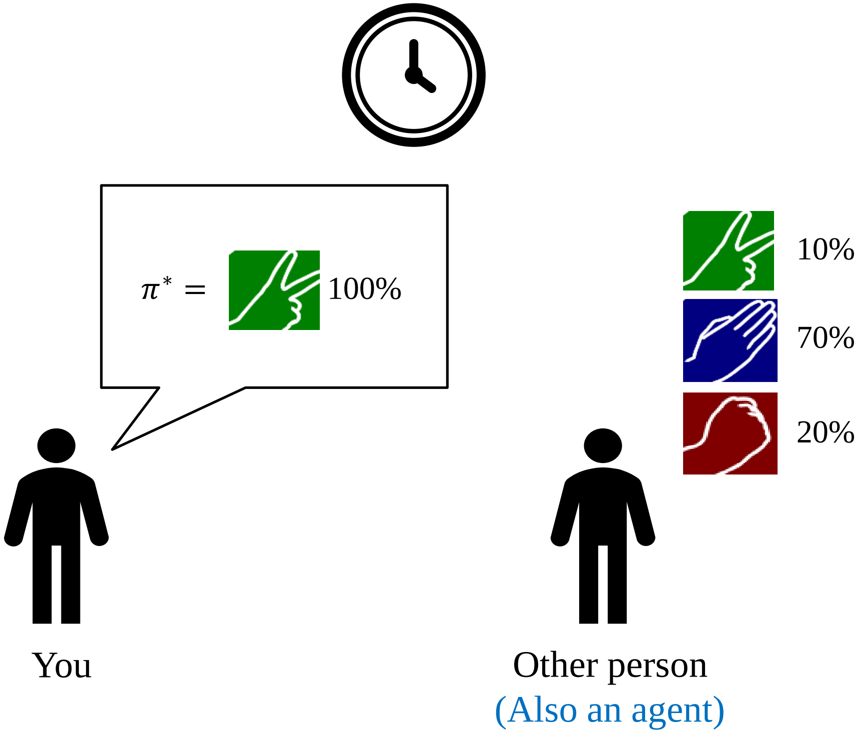
Stationary Environment

Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Multi-Agent RL

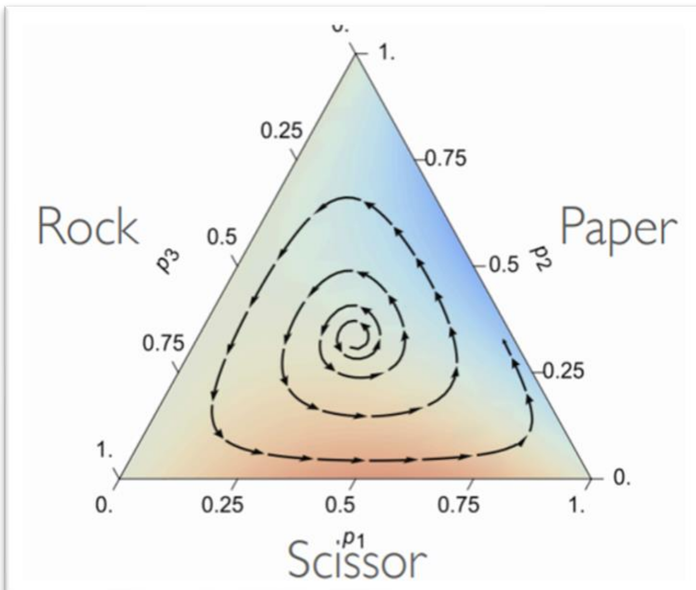


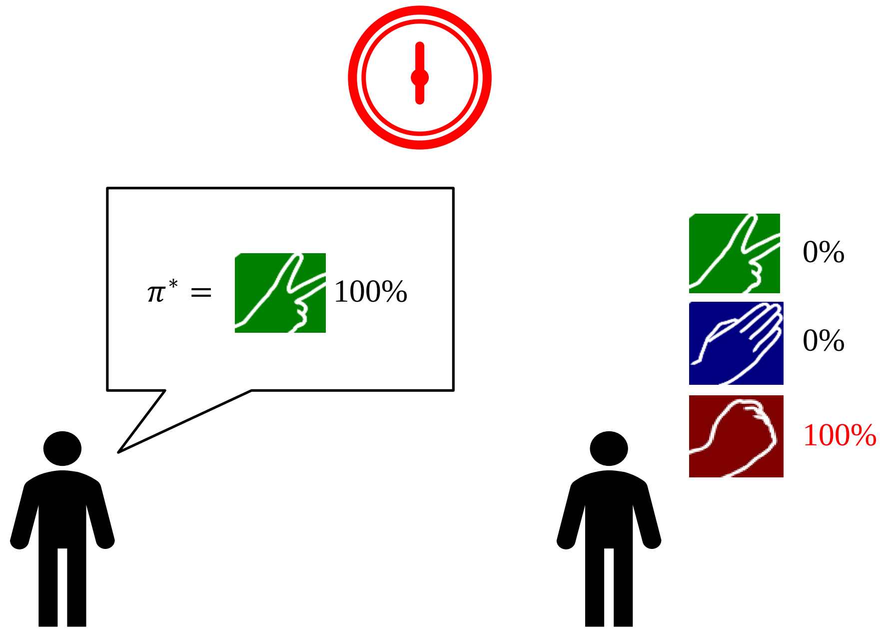


Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Multi-Agent RL





You

Other person
(Also an agent)

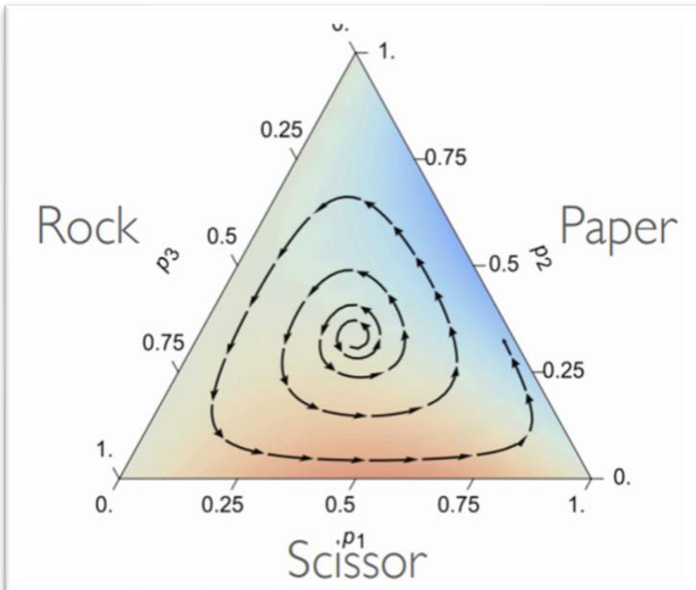
The person is learning!
It changes its strategy!

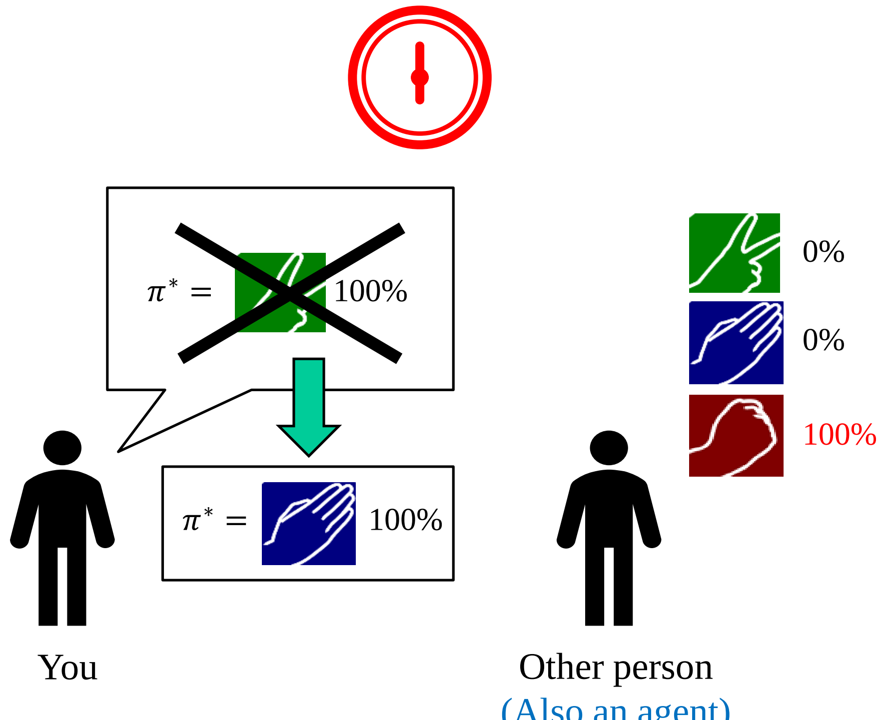
I-Chen Wu

Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Multi-Agent RL





The objective keeps changing,
and so does the optimal policy!

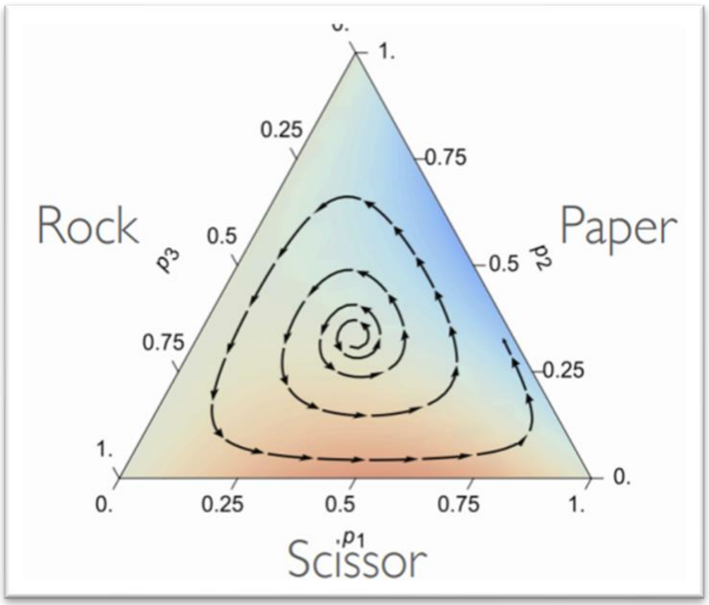
The person is learning!
It changes its strategy!

I-Chen Wu

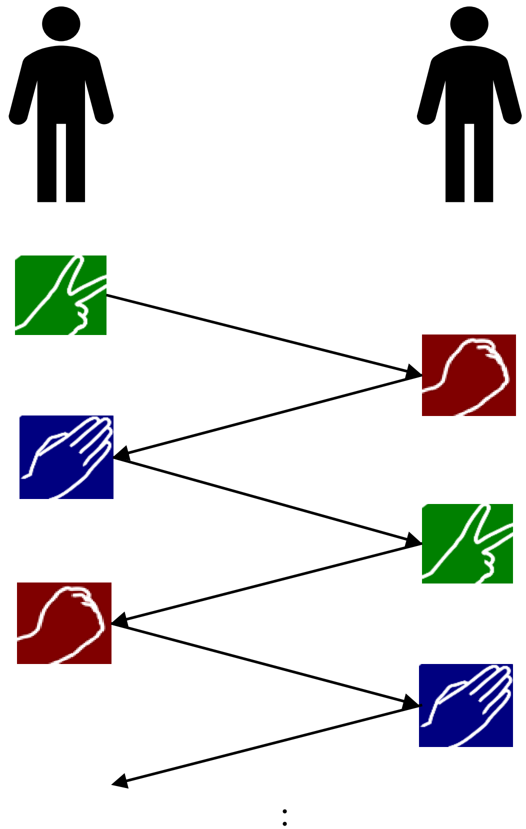
Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Multi-Agent RL



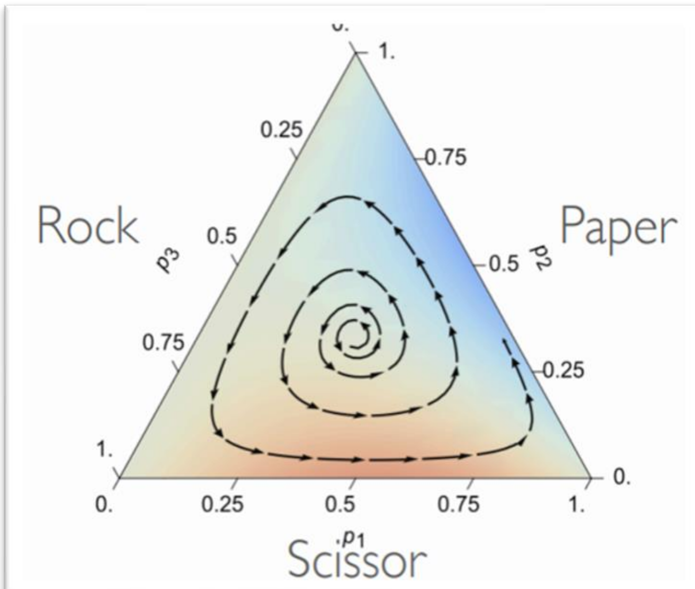
You Other person

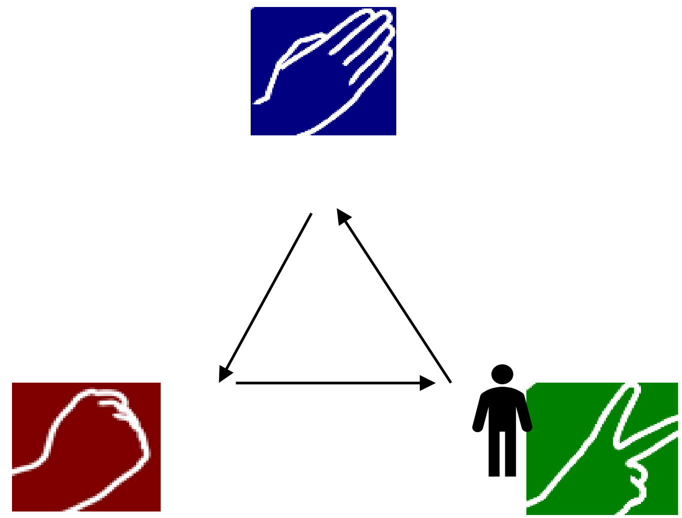


Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Multi-Agent RL

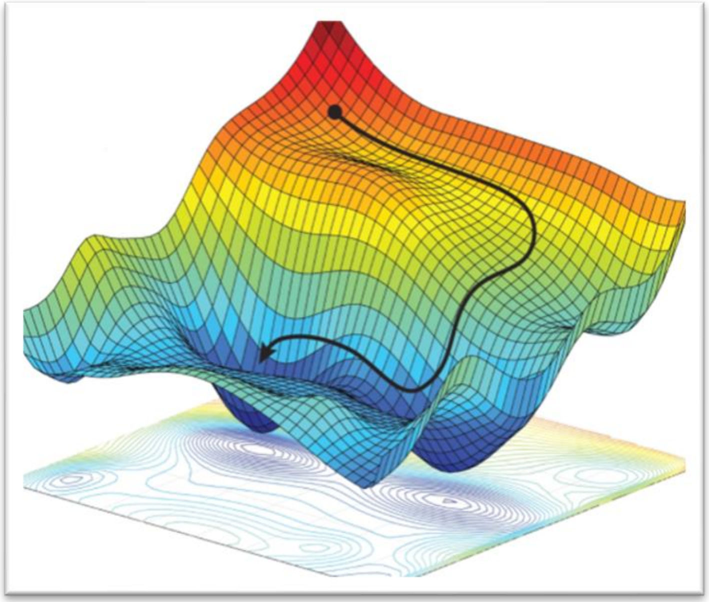




Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Single-Agent RL



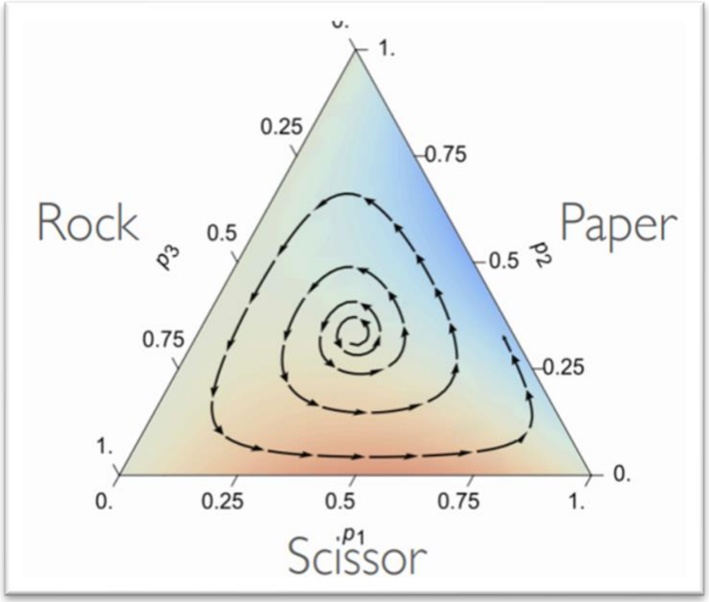
Objective of an agent:

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

Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

Multi-Agent RL



Objective of an agent:

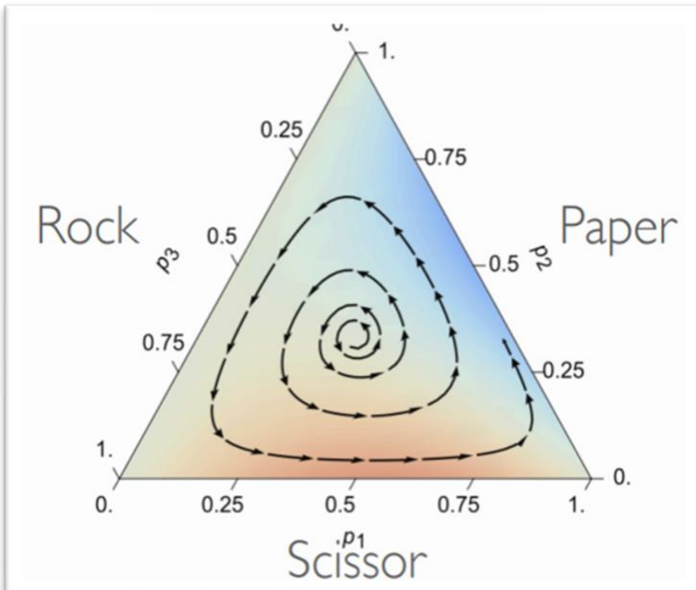
$$\max_{\pi}\left(\mathbb{E}_{\substack{a_t\sim\pi(\cdot|s_t),\\ \boxed{s_{t+1}\sim P(\cdot|s_t,a_t)}}}\left[\sum_{t=0}^{\infty}\gamma^t \boxed{R(s_t,a_t)}\right]\right)$$

The transition and reward functions is non-stationary!

Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

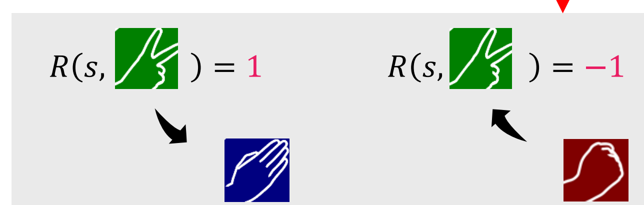
Non-Stationarity

Multi-Agent RL



Objective of an agent:

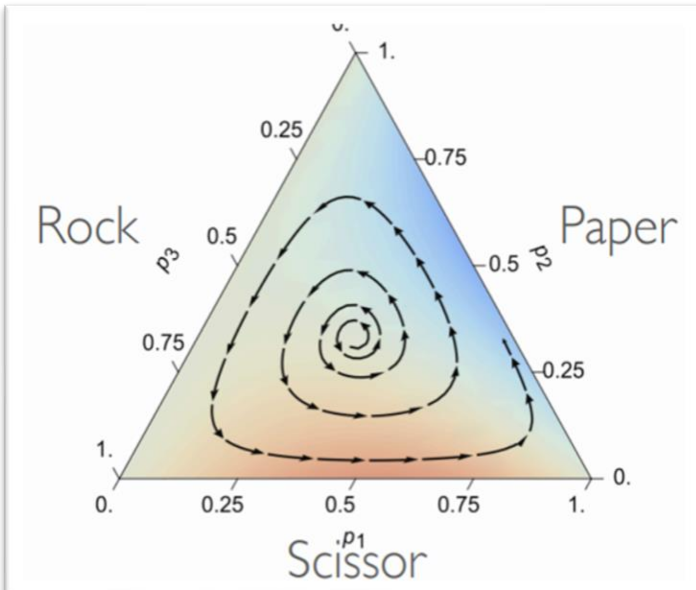
$$\max_{\pi}\left(\mathbb{E}_{\substack{a_t\sim\pi(\cdot|s_t),\\ \boxed{s_{t+1}\sim P(\cdot|s_t,a_t)}}}\left[\sum_{t=0}^{\infty}\gamma^t \boxed{R(s_t,a_t)}\right]\right)$$



Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

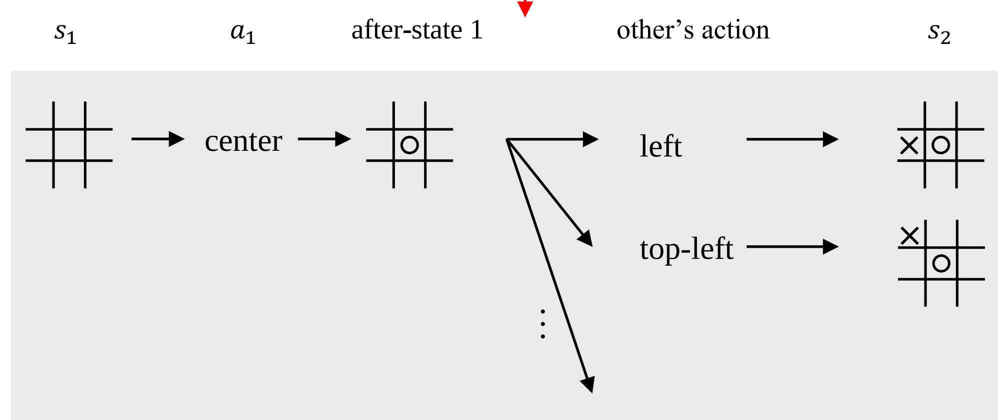
Non-Stationarity

Multi-Agent RL



Objective of an agent:

$$\max_{\pi}\left(\mathbb{E}_{\substack{a_t\sim\pi(\cdot|s_t),\\ \boxed{s_{t+1}\sim P(\cdot|s_t,a_t)}}}\left[\sum_{t=0}^{\infty}\gamma^t \boxed{R(s_t,a_t)}\right]\right)$$



Next state is influenced by other's policy
And other's policy is changing!

Source of images: (1) [gcoop] (2) https://en.wikipedia.org/wiki/Rock_paper_scissors

Non-Stationarity

- Dynamic loss (objective) landscape
  - Agents need to take into account other learning agents
  - Can harm the stationarity assumption for the theoretical guarantee of single-agent RL methods!
- Challenging to interpret when learning
  - Why do I get a lower reward by the same action in the same state?
    - Due to environment randomness?
    - Due to other agents

Complexity

- In MARL, each agent has to consider the actions of other agents to determining the best strategy
  - Joint action space $|\mathbb{A}|^N$ (N agents)
  - Joint action space grows exponentially with N

Case: 2 agents

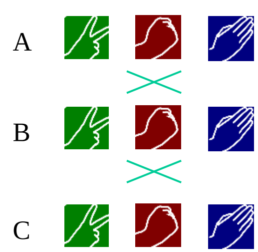
Agent B

Agent A

0,0	-1,1	1,-1
1,-1	0,0	-1,1
-1,1	1,-1	0,0

$|A|^N = 3^2$

Case: 3 agents



$|A|^N = 3^3$

…

Case: N agents

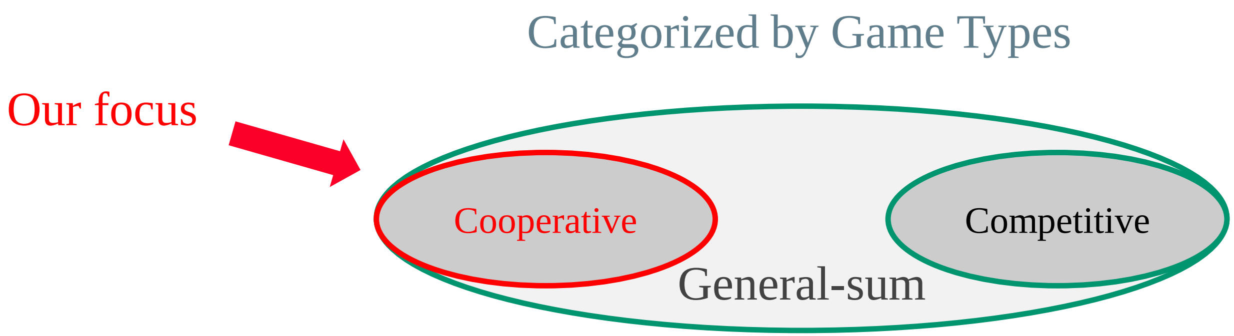
$|A|^N = 3^N$

I-Chen Wu

Introduction to MARL

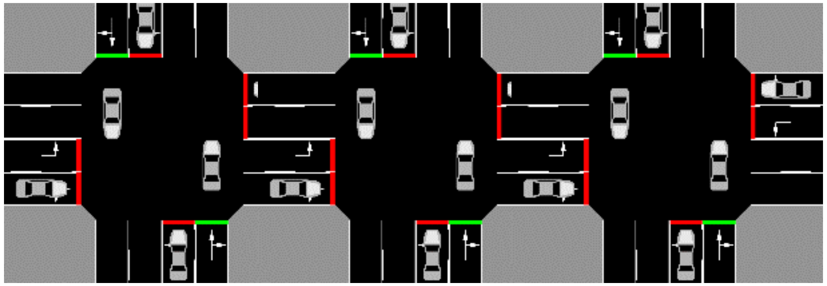
- Introduction to MARL
- Challenges of MARL
- Cooperative MARL

Categories of MARL



Case Study: Traffic Signal Control (TSC)

- Controllable components:
  - Traffic lights
- Objective
  - Minimize:
    - Waiting time
    - Travel time
    - …
  - Maximize
    - Throughput
    - …

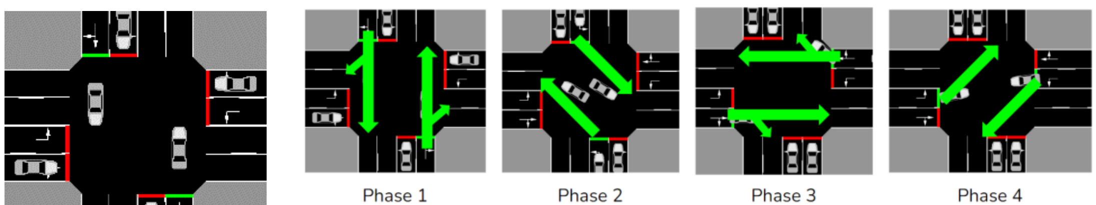


Case Study: Traffic Signal Control (TSC)

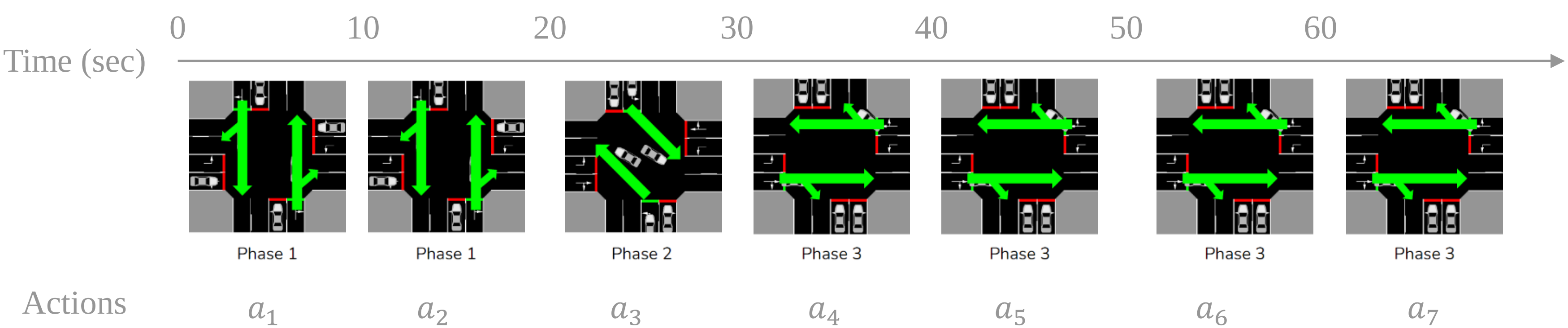
- Definitions of "action" for an intersection
  - Select next phase periodically
    - It will be used as examples for the following content
  - Determine next phase duration (the phase sequence is pre-defined)
  - …

Case Study: Traffic Signal Control (TSC)

- Example:
  - Action definition: select next phase every 10 seconds

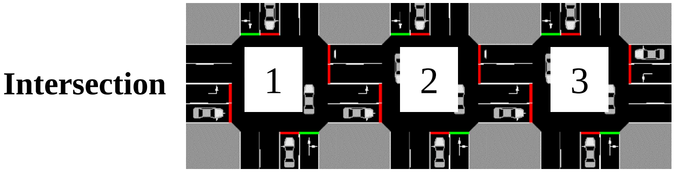


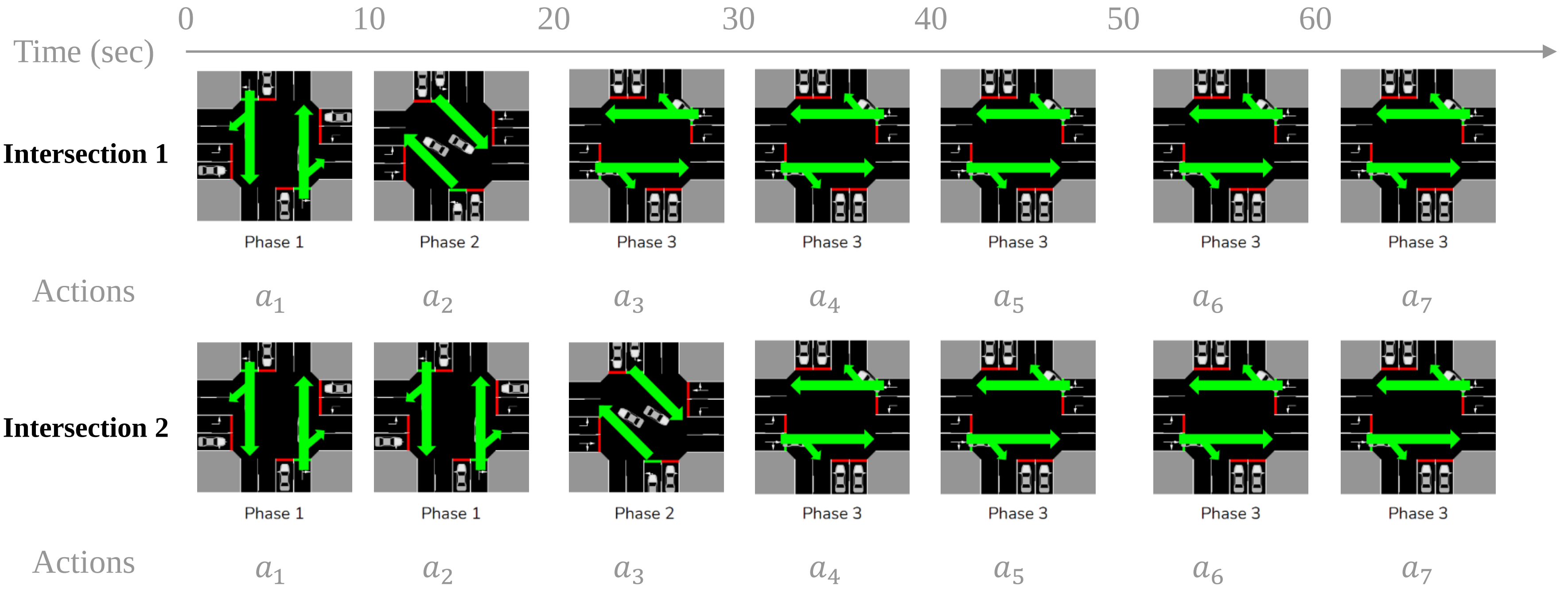
Assume 4 phases in total



Case Study: Traffic Signal Control (TSC)

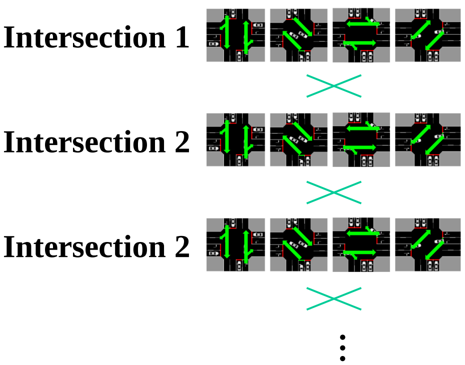
- Multiple intersections

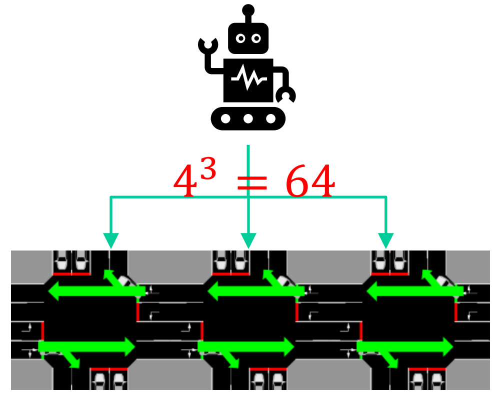


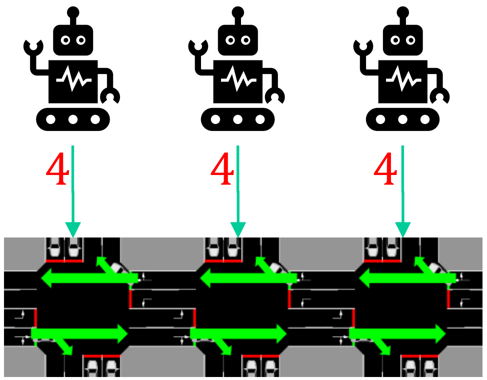


Case Study: Traffic Signal Control (TSC)

- Centralized control (single-agent):
  - Control all at a time
    - Joint action space: $|\mathcal{A}|^{N}$
- Decentralized (multi-agent):
  - One agent serves an intersection
    - Each agent's action space: $|\mathcal{A}|$ → smaller action space & more scalable
  - Agents learn how to cooperate to improve the traffic





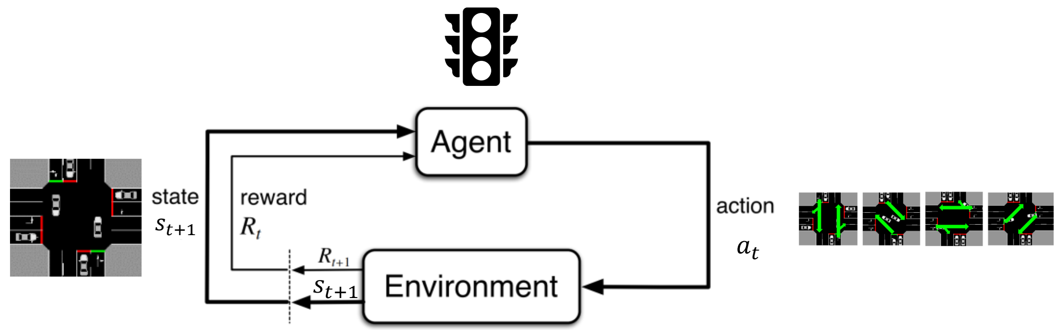


Cooperative MARL: Problem Formulation

- All agents receive the same reward as a team
- Assume all agents act together each timestep

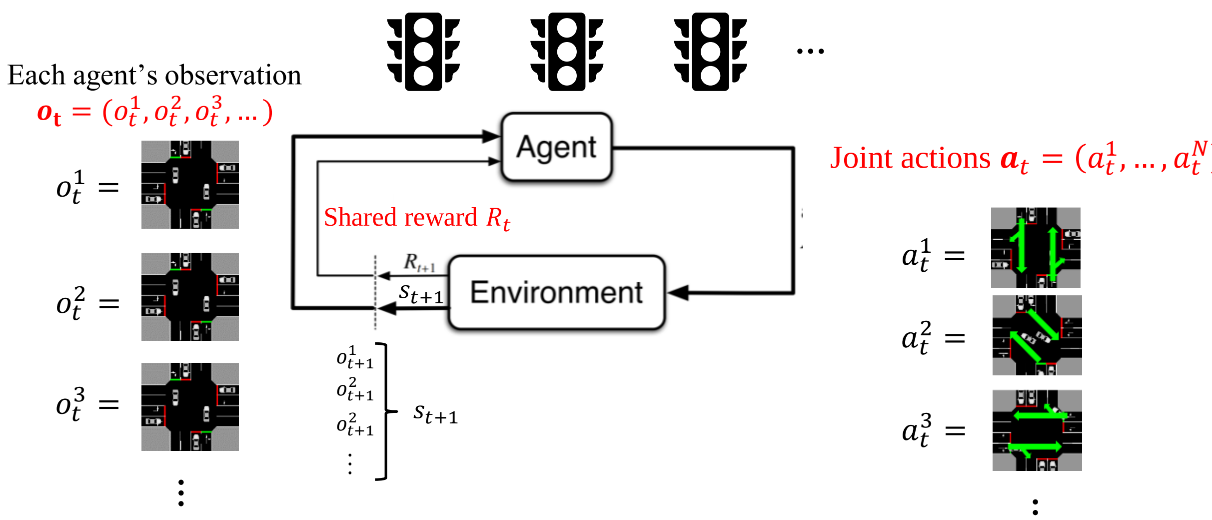
Problem Formulation (Single vs Multi Agent)

Single-Agent



Controllers for multiple intersections

Cooperative Multi-Agent



Cooperative MARL: Problem Formulation

- Consider a Partially-Observable Markov game: $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{Z}, P, \mathcal{R}, \gamma \rangle$
  - $t \in \mathbb{Z}^+$: times step
    - All agents act simultaneously at each time step
  - $\mathcal{N} = \{1, \dots, N\}$
    - $\mathcal{N}$: the set of agents
    - N: number of agents
    - Agent $i \in \mathcal{N}$
  - $s \in \mathcal{S}$
    - $\mathcal{S}$: state space
    - s: global environment state
    - s_t: global environment state at time step t
  - $\boldsymbol{a} \coloneqq \left(a^i\right)_{i \in \mathcal{N}} \in \mathcal{A}^N$
    - $\boldsymbol{a}$: joint action
    - $\mathcal{A}$: action space
    - a^i: agent i's action
  - $o^i = \mathcal{Z}(s; i)$
    - $\mathcal{O}$: Local observation space
    - $o^i \in \mathcal{O}$: agent i's local observation
    - $\mathcal{Z}$: observation function
  - Agents optimize towards one shard reward: $\mathcal{R}(s, \boldsymbol{a}): \mathcal{S} \times \mathcal{A}^N \to \mathbb{R}$
  - $\tau^i \in \mathcal{T} \coloneqq (\mathcal{O} \times \mathcal{A})^t$
    - τ^i: agent i's action-observation
    - $\boldsymbol{\tau} \coloneqq \left(\tau^i\right)_{i \in \mathcal{N}} \in \mathcal{T}^N$: all of the agent histories
  - For $a, \boldsymbol{a}, o, \boldsymbol{o}, \tau, \boldsymbol{\tau}$, etc.:
    - With subscript t: at time step t
    - With superscript i: of agent i
    - E.g. a_t^i means the action of agent i at time step t
  - $P(s' | s, \boldsymbol{a}) \coloneqq \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \to [0,1]$: state transition dynamics
  - $\gamma \in [0,1)$: discount factor

Problem Formulation and Notation

- $\boldsymbol{\pi} \coloneqq (\pi^i)_{i \in \mathcal{N}}$
  - $\boldsymbol{\pi}$: joint policy
  - π^i: agent i's policy
  - Different policy settings
    - $\pi^i(a^i|\tau^i): \mathcal{T} \times \mathcal{A} \to [0,1]$
    - $\pi^i(a^i|o^i): \mathcal{O} \times \mathcal{A} \to [0,1]$
    - $\pi^i(a^i|s): \mathcal{S} \times \mathcal{A} \to [0,1]$
- $G_t \coloneqq \sum_j^\infty \gamma^j R_{t+i}$: total accumulative rewards
- $\rho_{\boldsymbol{\pi}}(s) \coloneqq \sum_{t=0}^\infty \gamma^t \rho_{\boldsymbol{\pi}}^t(s)$
  - $\rho_{\boldsymbol{\pi}}(s)$: improper marginal state distribution
  - $\rho_{\boldsymbol{\pi}}^t(s)$: marginal state distribution at time t under joint policy $\boldsymbol{\pi}$
  - $\rho^0(s)$: initial state distribution
- Value function
  - Use $u/\boldsymbol{u}$ to denote the input instance of $a/\boldsymbol{a}$
  - $V_{\boldsymbol{\pi}}(s) \coloneqq \mathbb{E}_{\boldsymbol{a}_{0:\infty}\sim\boldsymbol{\pi}, s_{1:\infty}\sim\mathcal{P}}[\sum_{t=0}^\infty \gamma^t R_t | s_0 = s]$
  - $Q_{\boldsymbol{\pi}}(s,\boldsymbol{u}) \coloneqq \mathbb{E}_{s_{1:\infty}\sim\mathcal{P}, \boldsymbol{a}_{1:\infty}\sim\boldsymbol{\pi}}[\sum_{t=0}^\infty \gamma^t R_t | s_0 = s, \boldsymbol{a}_0 = \boldsymbol{u}]$
  - $A_{\boldsymbol{\pi}}(s,\boldsymbol{u}) \coloneqq Q_{\boldsymbol{\pi}}(s,\boldsymbol{u}) - V_{\boldsymbol{\pi}}(s)$
- Consider a fully-cooperative setting where all agents share the same reward function, aiming to maximize the expected total reward:

$$J(\boldsymbol{\pi}) \coloneqq \mathbb{E}_{s_{0:\infty}\sim\rho_{\boldsymbol{\pi}}^{0:\infty}, \boldsymbol{a}_{0:\infty}\sim\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$$

State, Observation, and History

- The usage depends on your scenarios
  - State s:
    - Global state
  - Observation o:
    - Maybe partial observation or global state
  - History τ:
    - Often used in partial observable settings
    - Try to recover the state by history

Value-based Cooperative MARL

- Centralized Training Decentralized Execution (CTDE)
- Value-Decomposition Network (VDN)
- QMIX

Value-based Cooperative MARL

- Centralized Training Decentralized Execution (CTDE)
- Value-Decomposition Network (VDN)
- QMIX

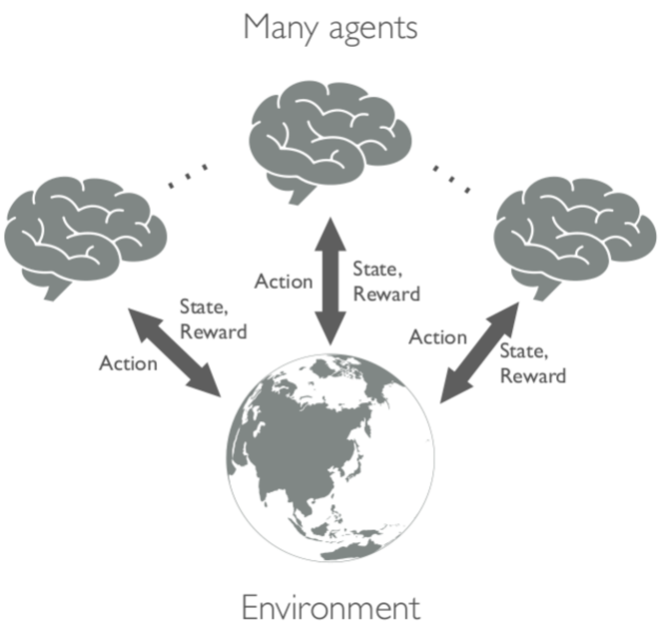
Recall: Q-Learning in Single-Agent RL

1. Learn optimal state-action value function Q^*

$$Q^{(new)}(s_t, a_t) = Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q(s_{t+1}, a)\right))$$

2. Derive action by

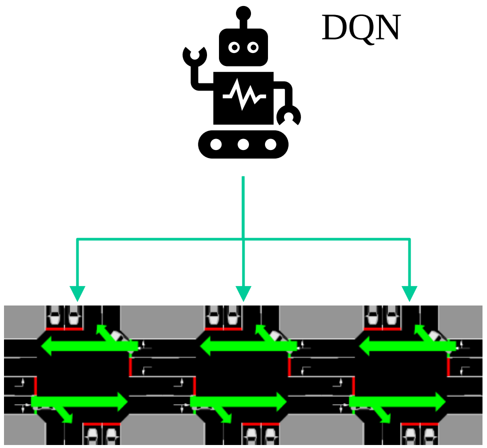
$$\arg\max_a Q^*(s, a)$$

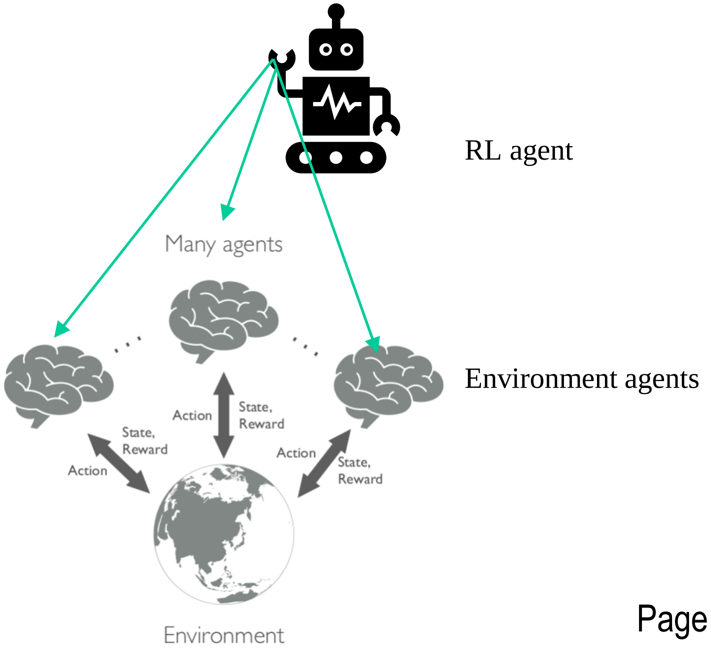


How about Q-learning in Cooperative MARL?

Centralized Control

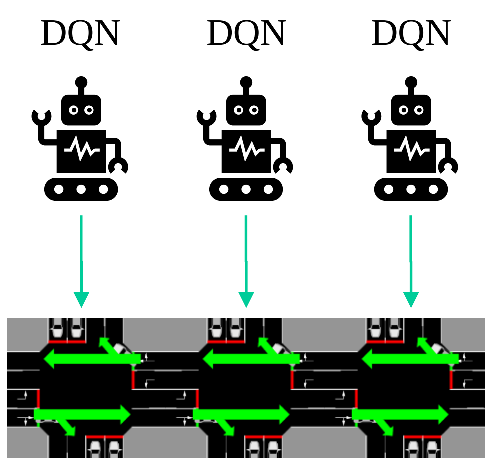
- Straightly use Single-Agent Q-learning by centralized control
  - An RL agent controls all environment agents
- Issue: curse of dimensions of joint action space

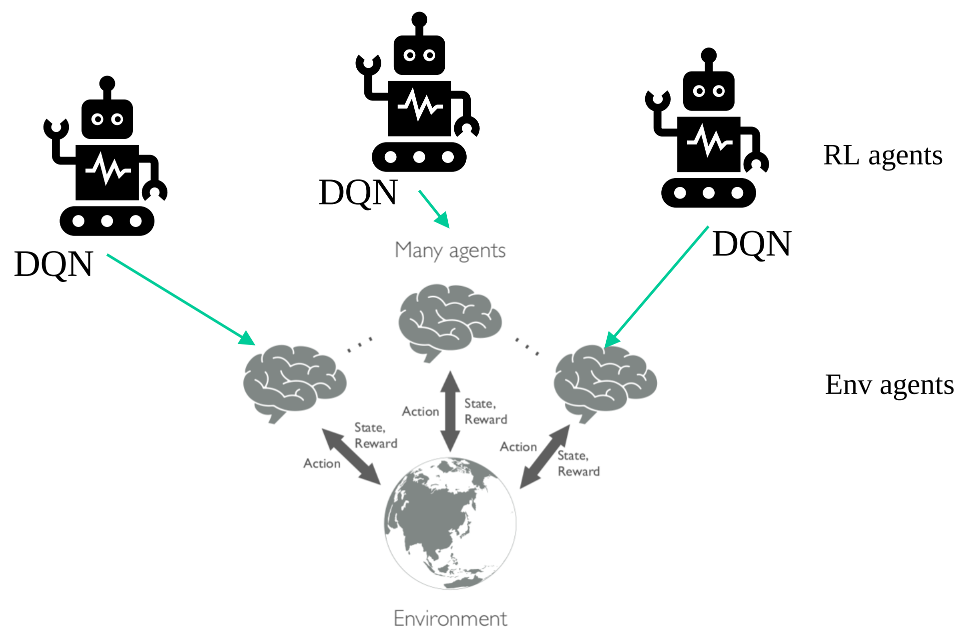




Independent Control (Decentralized)

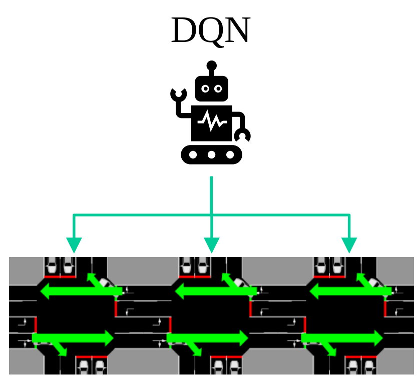
- Straightly apply Single-Agent Q-learning on each agent independently
  - Each agent views other agents as a part of environment
- Issue: non-stationarity

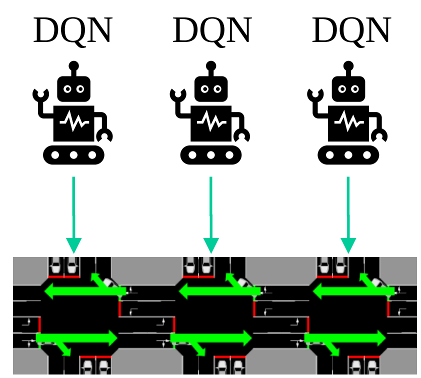




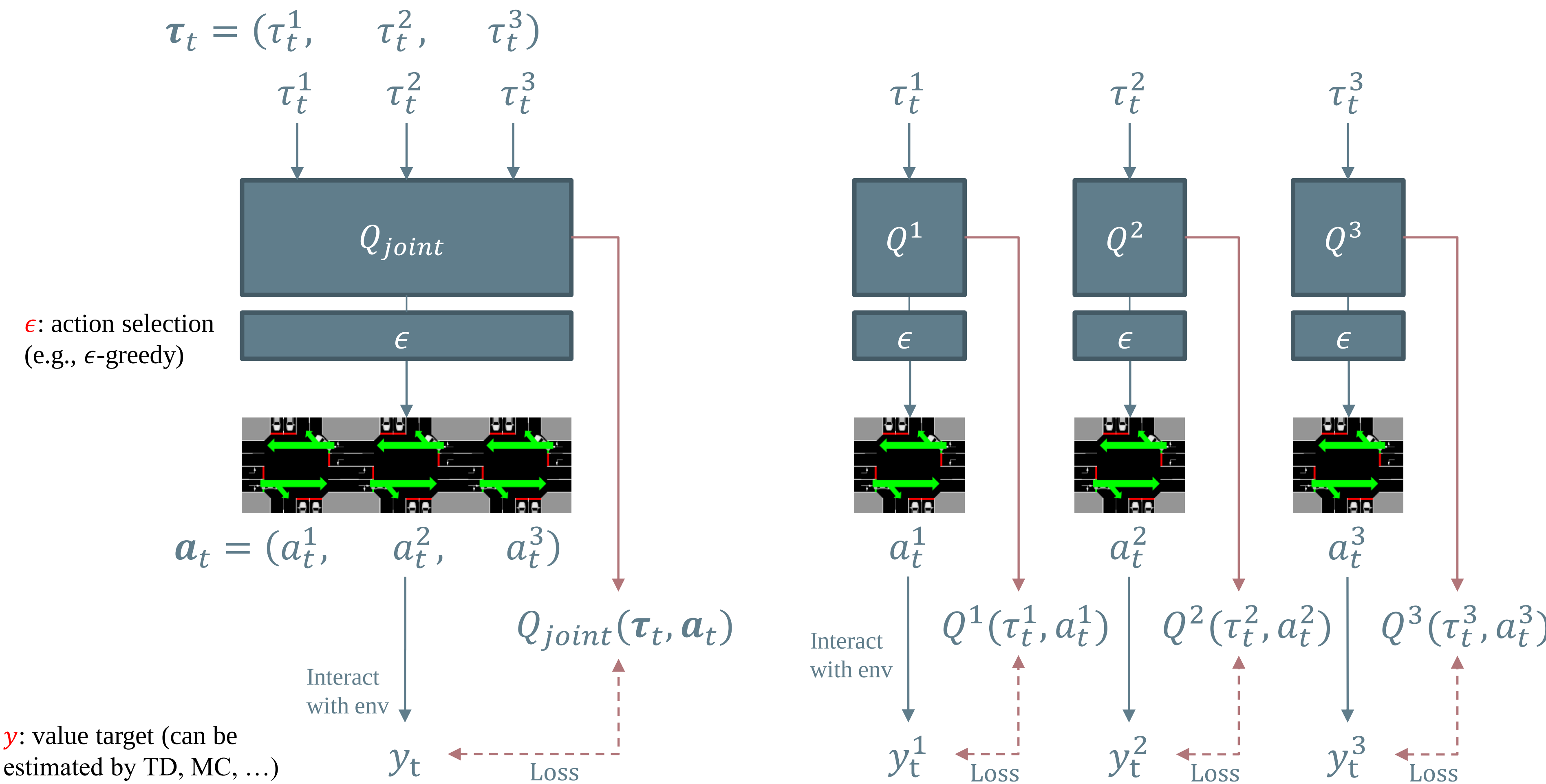
Centralized Control vs Independent Control

	Centralized Control	Independent Control
Pros	✓ Direct use of single-agent methods ✓ Preserving single-agent methods' theoretical nature	✓ Direct use of single-agent methods ✓ Smaller action space
Cons	× Large joint action space	× Non-stationary; harming theoretical guarantee





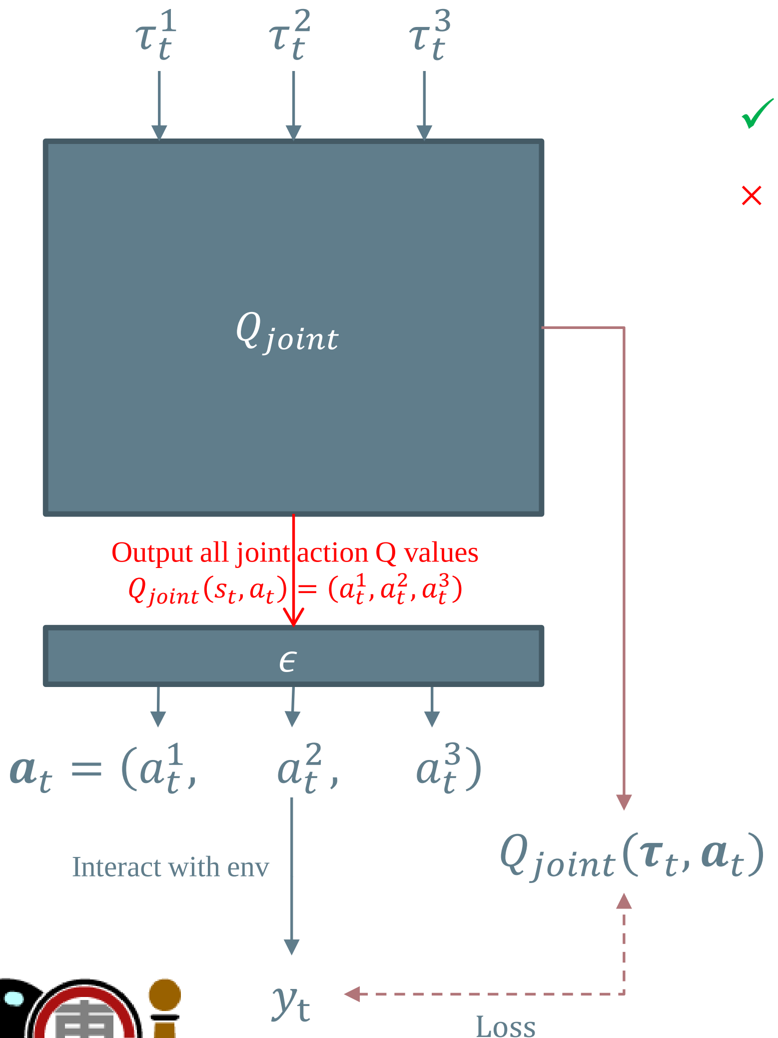
Centralized Control vs Independent Control



Centralized Control **Independent Control**

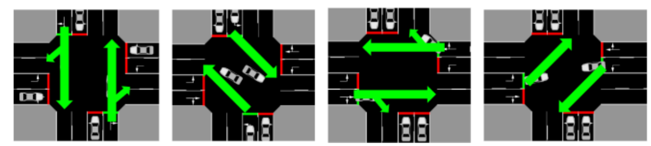
I-Chen Wu

Centralized Control: Advantage & Disadvantage



✓ Preserving single-agent methods' theoretical nature

× Large joint action space



Phase 1 2 3 4

Action space size of an intersection $|\mathcal{A}| = 4$

$Q_{joint}(\boldsymbol{\tau}_t, \quad a_t^1, \quad a_t^2, \quad a_t^3 \quad)$

$$\left.\begin{array}{l} Q_{joint}(\boldsymbol{\tau}, \text{phase1}, \text{phase1}, \text{phase1}) \\ Q_{joint}(\boldsymbol{\tau}_t, \text{phase1}, \text{phase1}, \text{phase2}) \\ Q_{joint}(\boldsymbol{\tau}_t, \text{phase1}, \text{phase1}, \text{phase3}) \\ \vdots \\ Q_{joint}(\boldsymbol{\tau}_t, \text{phase4}, \text{phase4}, \text{phase4}) \end{array}\right\} |\mathcal{A}|^3 = 64$$

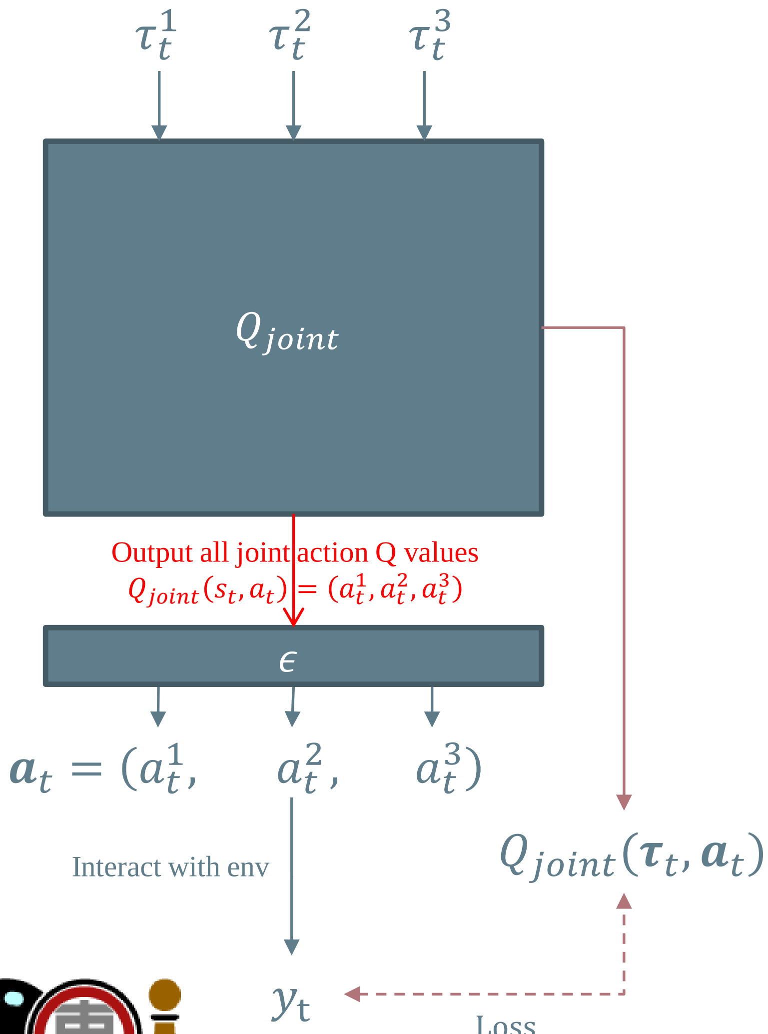
I-Chen Wu

Centralized Training Decentralized Execution (CTDE)

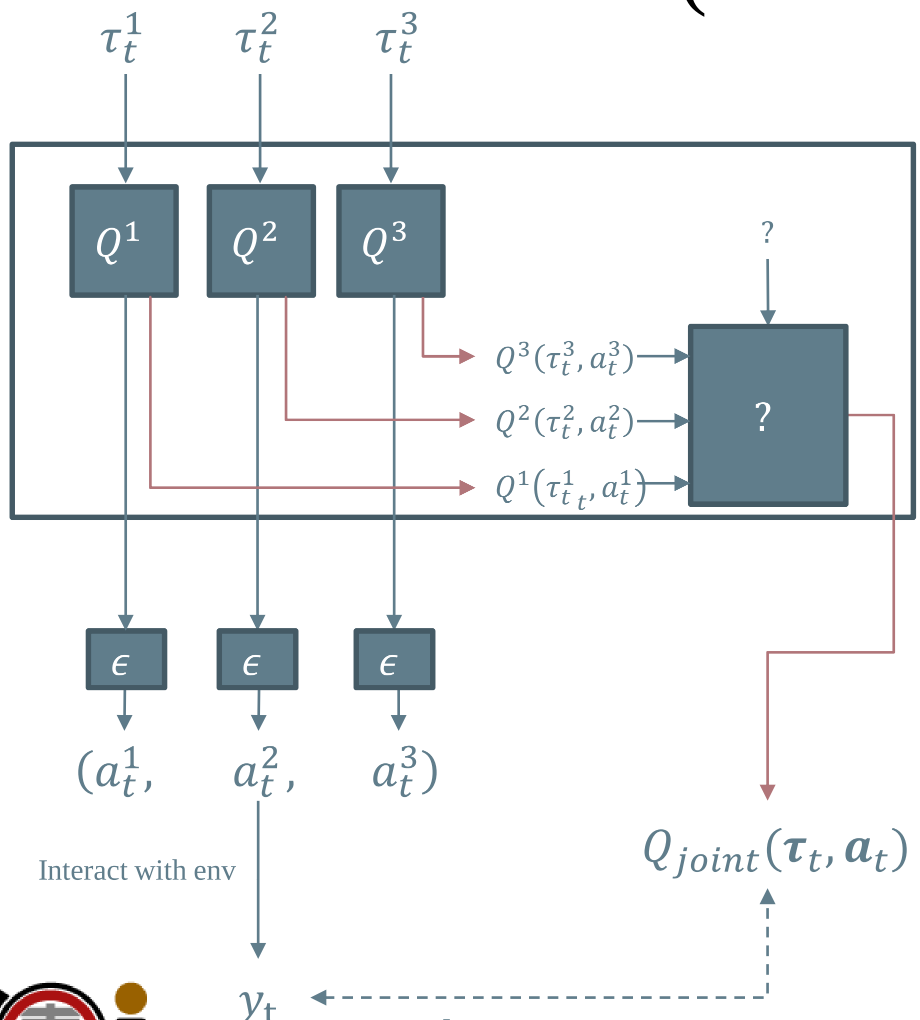
- CTDE: a common idea in value-based MARL
  - Even for policy-based MARL
- Centralized Training:
  - Have access to global information when training
- Decentralized Execution:
  - Agents make their own decisions only based on decentralized local policies
  - Q-learning case:
    - Select an optimal action from individual local Q functions

[QTRAN]; Zhou, Yihe, et al. "Is Centralized Training with Decentralized Execution Framework Centralized Enough for MARL?." arXiv preprint arXiv:2305.17352 (2023).

Centralized Control

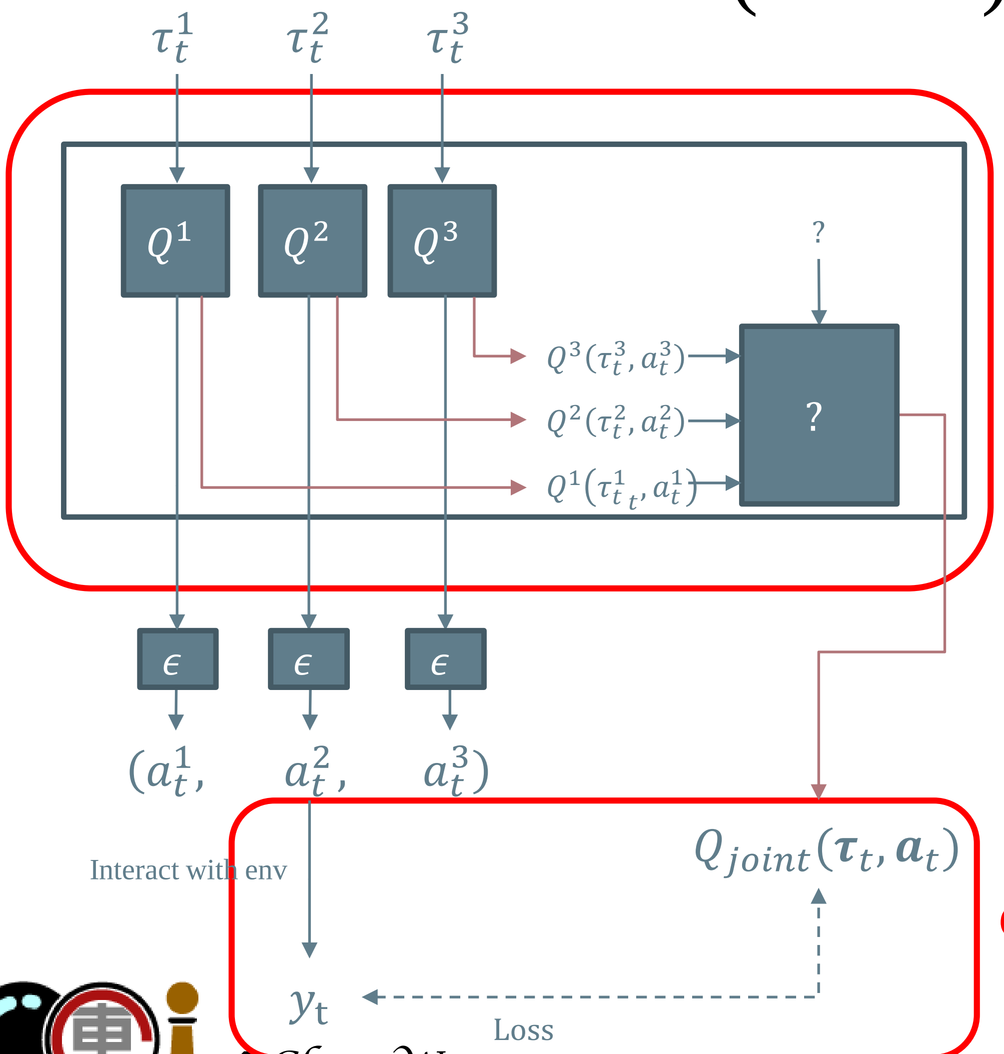


Centralized Training Decentralized Execution (CTDE)



- ✓ Preserving single-agent methods' theoretical nature
- × ~~Large joint action space~~
- ✓ Smaller action space

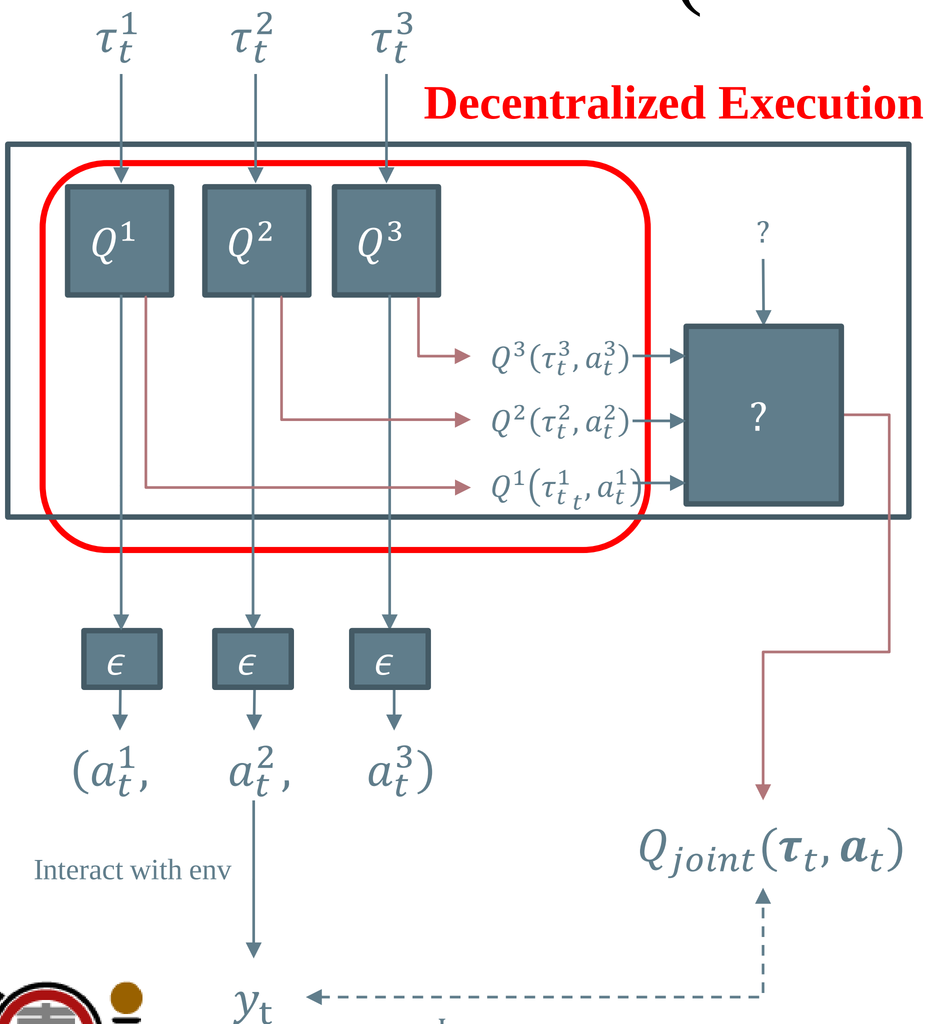
Centralized Training Decentralized Execution (CTDE)



- ✓ Preserving single-agent methods' theoretical nature
- × ~~Large joint action space~~
- ✓ Smaller action space

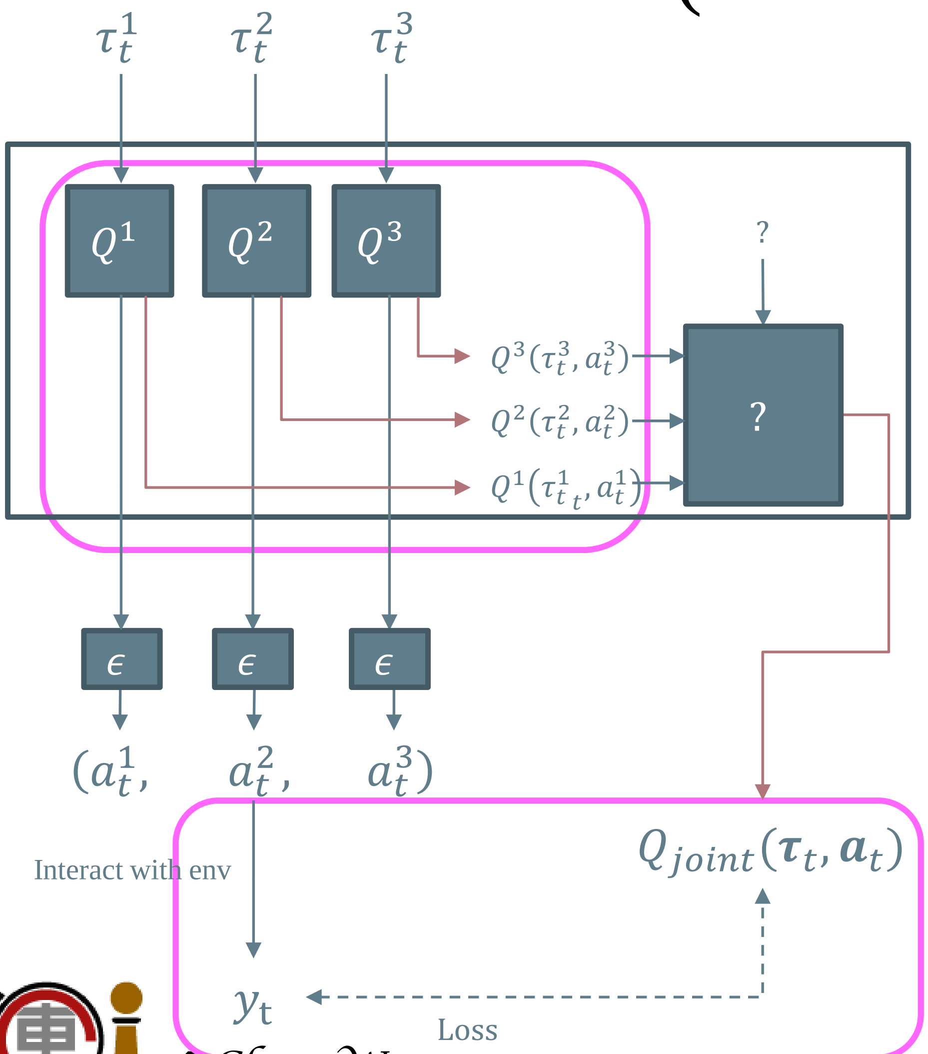
Centralized Training

Centralized Training Decentralized Execution (CTDE)



- ✓ Preserving single-agent methods' theoretical nature
- × ~~Large joint action space~~
- ✓ Smaller action space

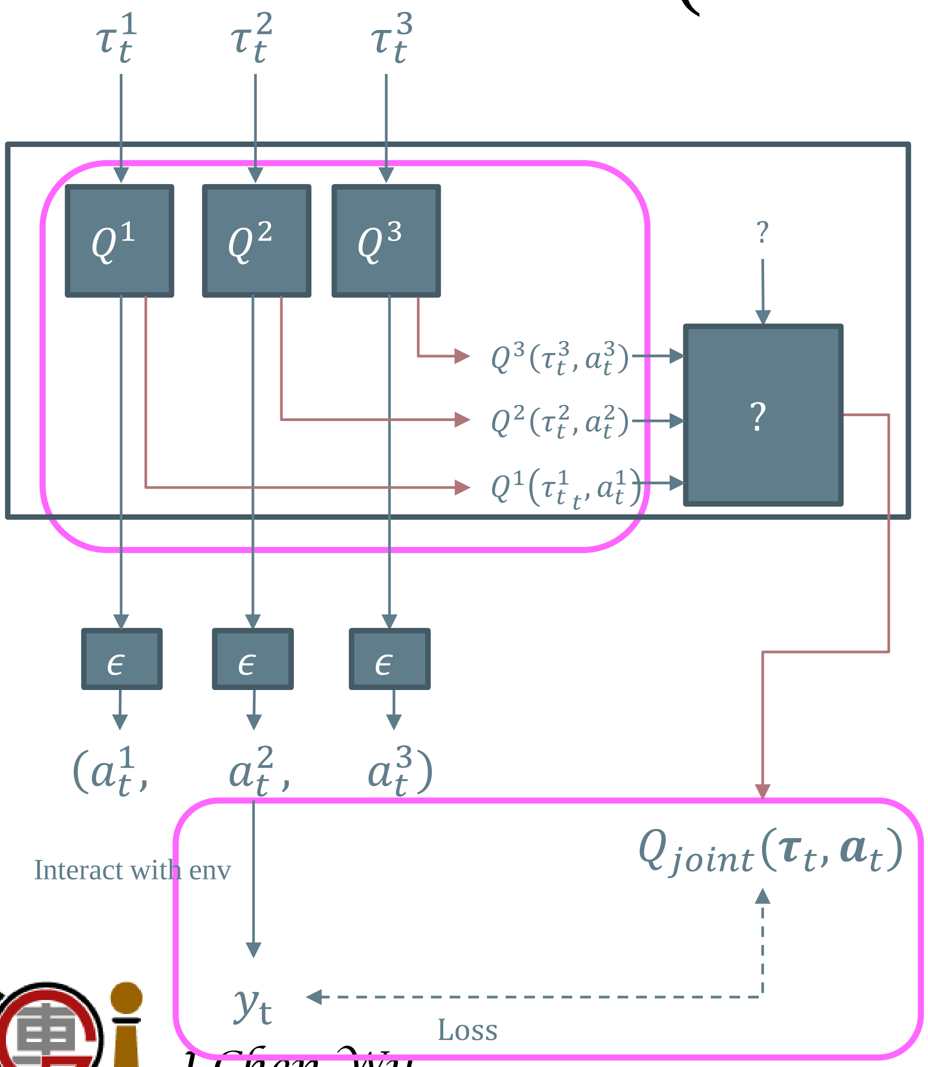
Centralized Training Decentralized Execution (CTDE)



- ✓ Preserving single-agent methods' theoretical nature
- × ~~Large joint action space~~
- ✓ Smaller action space

Learn an optimal Q_{joint}

Centralized Training Decentralized Execution (CTDE)

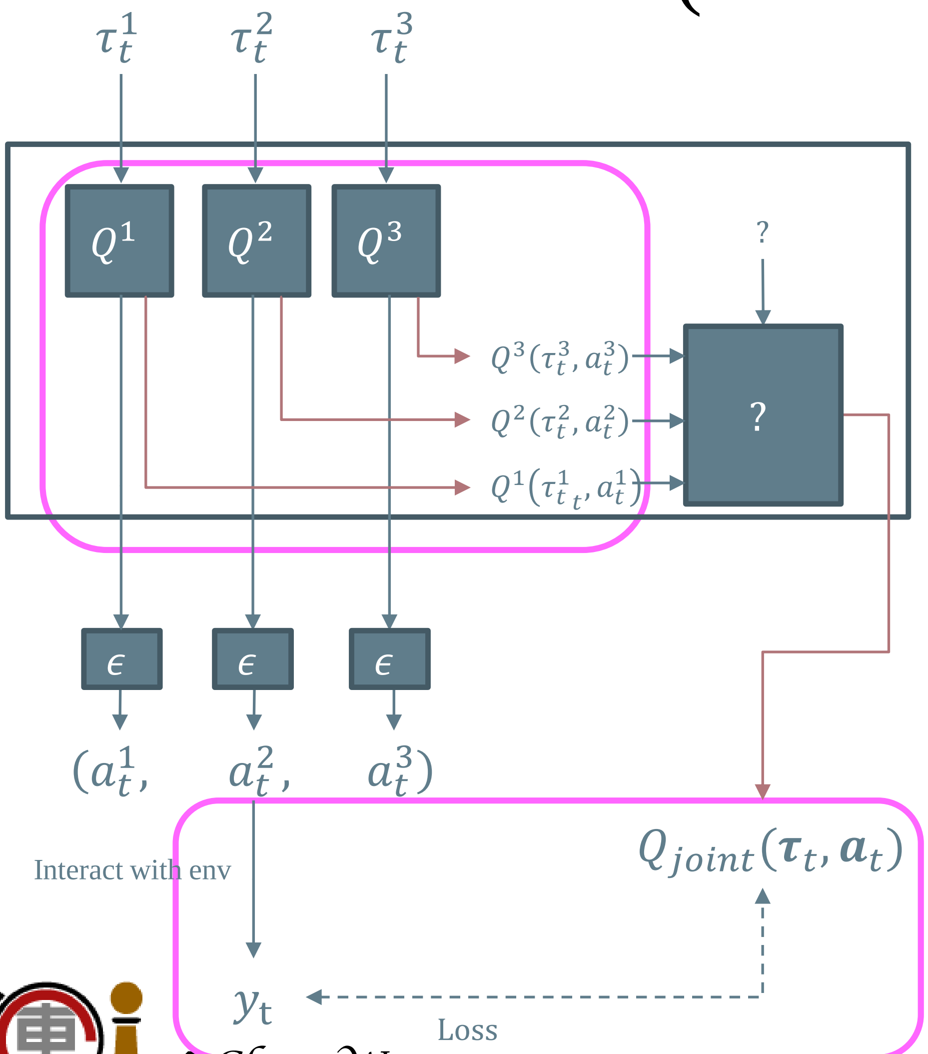


- ✓ Preserving single-agent methods' theoretical nature
- × ~~Large joint action space~~
- ✓ Smaller action space

But it takes actions from individual Q
Instead of Q_{joint}

Learn an optimal Q_{joint}

Centralized Training Decentralized Execution (CTDE)



- ✓ Preserving single-agent methods' theoretical nature
- × ~~Large joint action space~~
- ✓ Smaller action space

But it takes actions from individual Q
Instead of Q_{joint}

So it needs to ensure:

$$\begin{pmatrix} \arg\max_{a'} Q^1(\tau^1, a'), \\ \arg\max_{a'} Q^2(\tau^2, a'), \\ \arg\max_{a'} Q^3(\tau^3, a'), \end{pmatrix}$$

$$||$$

$$\arg\max_{\boldsymbol{a}'} Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}')$$

So called IGM next slide!

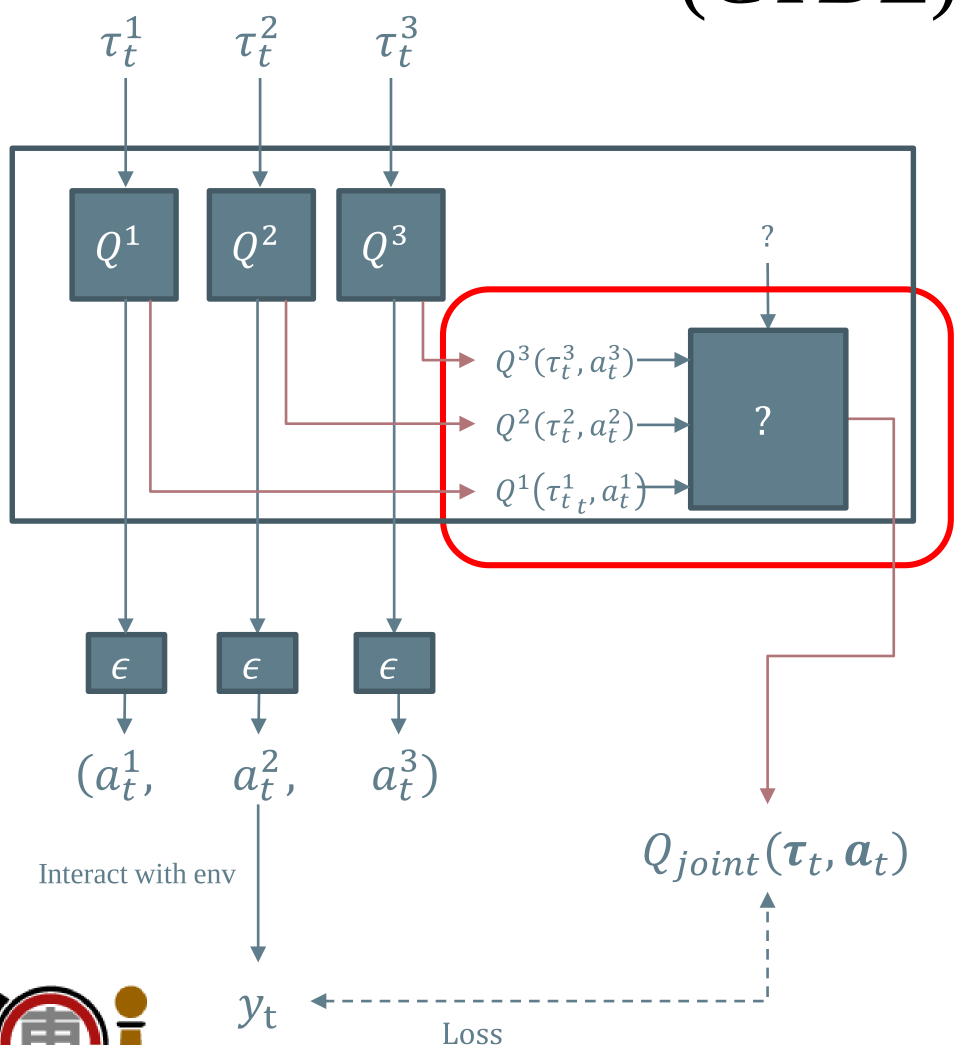
Learn an optimal Q_{joint}

Individual-Global-Maximum (IGM)

- Individual-Global-Maximum (IGM)
  - For a joint action-value function $Q_{joint}: \mathcal{T}^N \times \mathcal{A}^N \rightarrow \mathbb{R}$, where $\boldsymbol{\tau} \in \mathcal{T}$ is a joint action-observation histories, if there exist individual action-value function $\left[Q^i: \mathcal{T} \times \mathcal{A} \rightarrow \mathbb{R}\right]_{i=1}^{N}$, such that the following holds:

$$\arg\max_{\boldsymbol{a}} Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a_1} Q^1(\tau^1, a^1) \\ \arg\max_{a_2} Q^2(\tau^2, a^2) \\ \dots \\ \arg\max_{a_N} Q^N(\tau^N, a^N) \end{pmatrix}$$

Centralized Training Decentralized Execution (CTDE)



How to decompose joint Q into local Q while satisfy IGM?

→ Algorithm design

$$\begin{pmatrix} \arg\max_{a'} Q^1(\tau^1, a'), \\ \arg\max_{a'} Q^2(\tau^2, a'), \\ \arg\max_{a'} Q^3(\tau^3, a'), \end{pmatrix}$$

$$||$$

$$\arg\max_{\boldsymbol{a}'} Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}')$$

Value-based Cooperative MARL

- Centralized Training Decentralized Execution (CTDE)
- Value-Decomposition Network (VDN)
- QMIX

Value-Decomposition Network (VDN)

- Value-Decomposition Networks For Cooperative Multi-Agent Learning
  - Sunehag, Peter, et al. "Value-decomposition networks for cooperative multi-agent learning." arXiv preprint arXiv:1706.05296 (2017). (https://arxiv.org/abs/1706.05296)
  - DeepMind

Spurious Reward and Lazy Agent Problems

- Spurious reward signals
  - Due to partial observation
- Lazy agents
  - If there is another agent learn a useful policy, the agent may be discouraged from learning because:
    - Its exploration would hinder the agent

Value-Decomposition Network (VDN)

- Solution:
  - Each agent learns its contribution

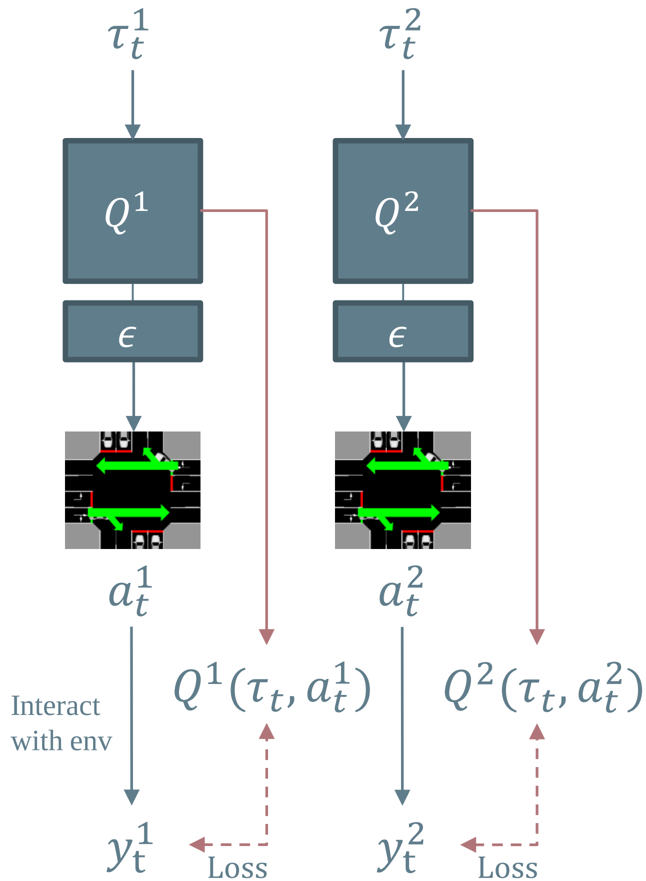
$$Q_{joint}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i, u^i)$$

- Assumption:
  - Q_{joint} can be additively decomposed

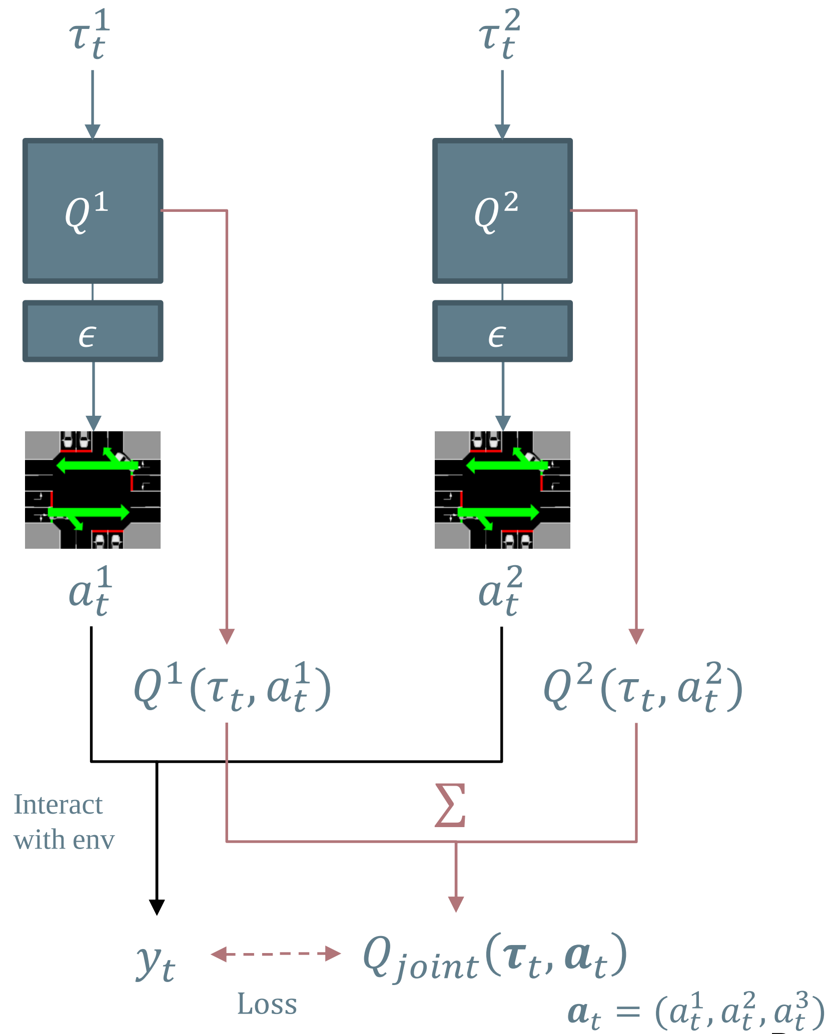
$$R(s, \boldsymbol{a}) = \sum_{i=1}^{N} r^i(\tau^i, a^i)$$

$$Q_{joint}(s, \boldsymbol{a}) \approx \sum_{i=1}^{N} Q^i(\tau^i, a^i)$$

Value-Decomposition Network (VDN)



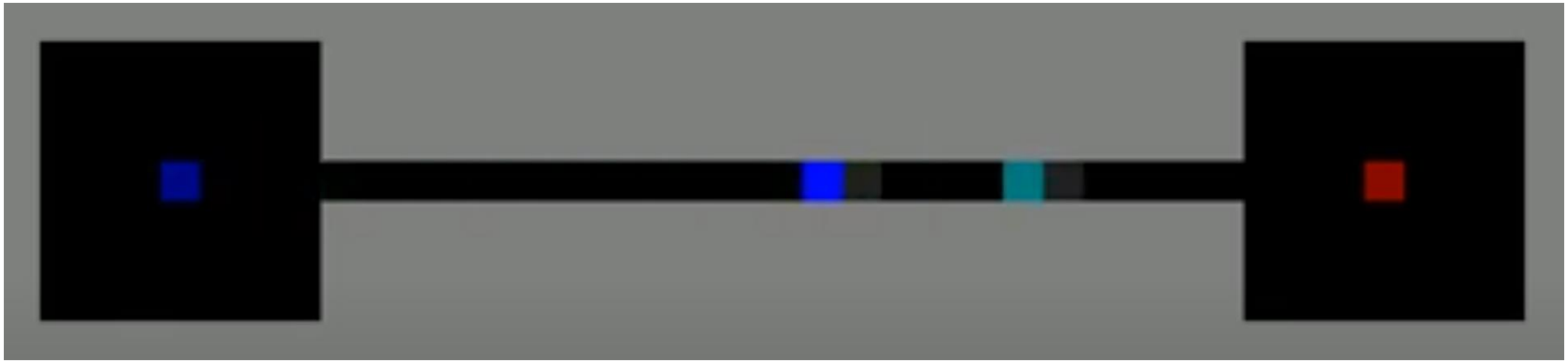
Independent Control



VDN

Example: Fetch Game

- Fetch game with one corridor
  - Two agents (agent 1 and agent 2)
  - Pick up the object (right side): +3, and then return to (left side): +5



Example: Fetch Game

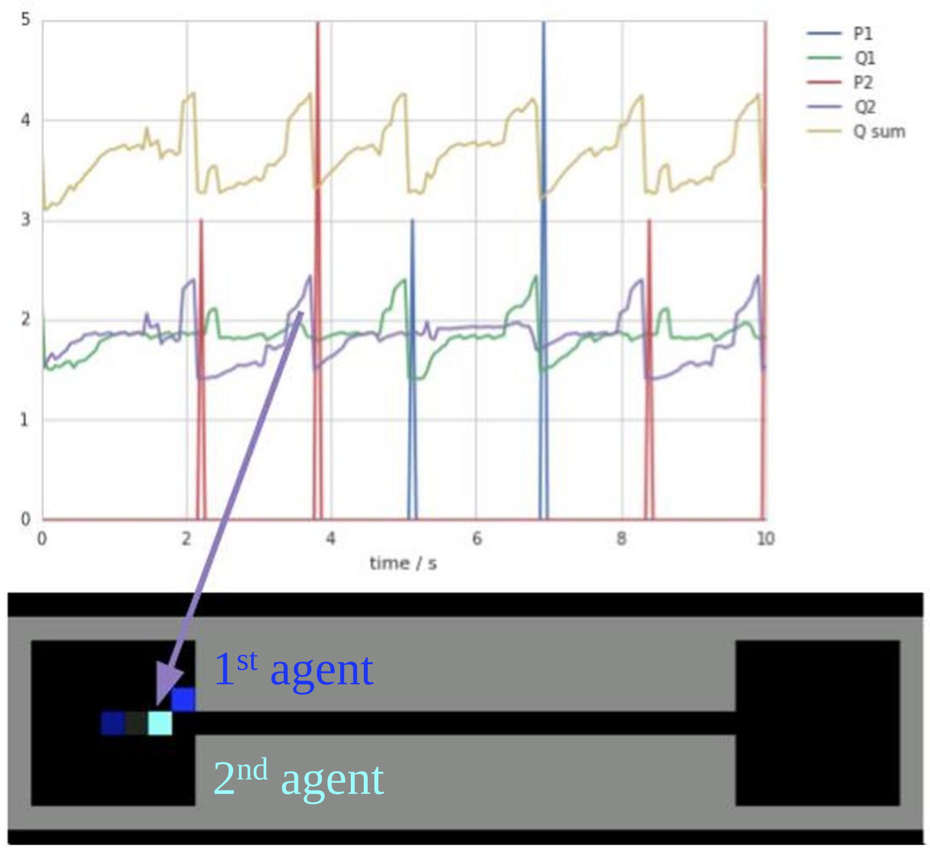
- Fetch video (VDN)

https://www.youtube.com/watch?v=aAH1eyUQsRo

Analyzing Q Values in Fetch

As blue agent (2nd agent) drops the object
Immediate reward occurs
→ Its Q value (Q2) spikes
→ The other agent's Q value (Q1) is flatten

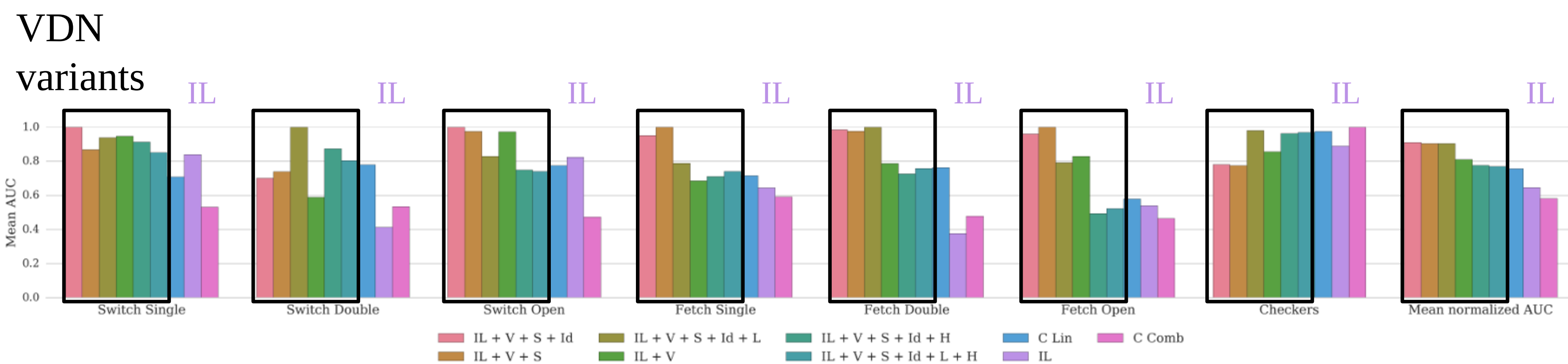
- **P1**: agent 1's immediate reward
- **Q1**: agent 1's Q-value
- **P2**: agent 2's immediate reward
- **Q2**: agent 2's Q-value
- Q sum: summation of their Q-values



[mc]

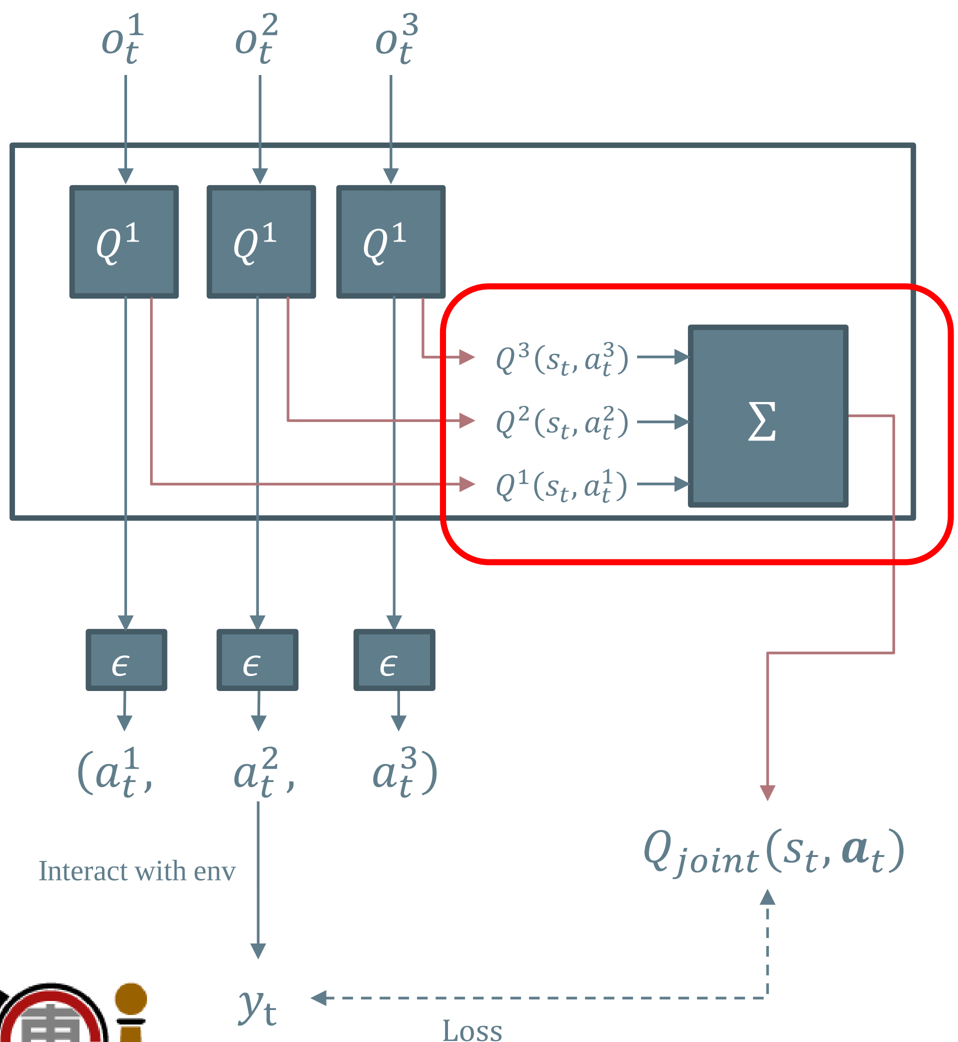
Experiment Results

- With VDN, the normalized AUC is better



[mc]

VDN (CTDE Framework)



VDN:

$$Q_{joint}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i, a^i)$$

$$\left.\begin{pmatrix} \arg\max_{a'} Q^1(\tau^1, a'), \\ \arg\max_{a'} Q^2(\tau^2, a'), \\ \arg\max_{a'} Q^3(\tau^3, a'), \end{pmatrix}\right] - \sum_i \max Q^i(\tau^i, a^i)$$

$$\| \qquad\qquad \|$$

$$\arg\max_{\boldsymbol{a}'} Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}') \qquad \max \sum_i Q^i(\tau^i, a^i)$$

$$\|$$

$$\max Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a})$$

I-Chen Wu

Value-based Cooperative MARL

- Centralized Training Decentralized Execution (CTDE)
- Value-Decomposition Network (VDN)
- QMIX

QMIX: Monotonic Value Function Factorisation

- Rashid, Tabish, et al. "Monotonic value function factorisation for deep multi-agent reinforcement learning." The Journal of Machine Learning Research 21.1 (2020): 7234-7284. (https://arxiv.org/abs/1803.11485)

Value-Decomposition Network (VDN)

- Solution:
  - Each agent learns its contribution

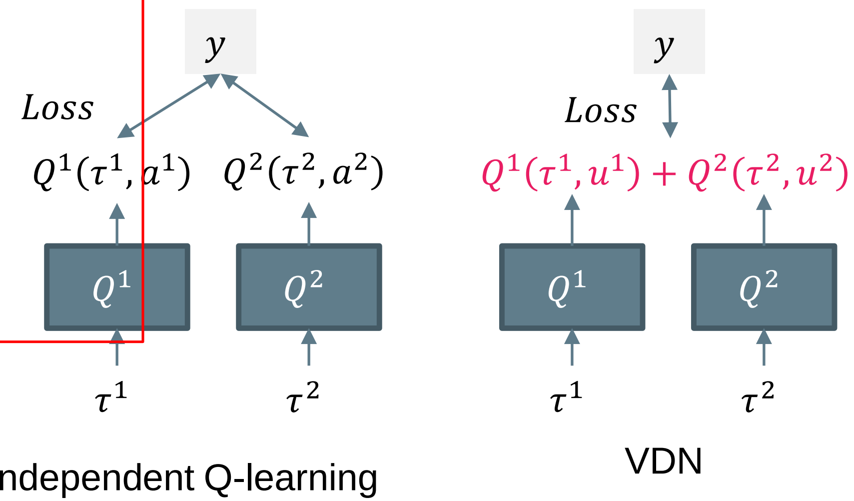
$$Q_{joint}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i, u^i)$$

- Assumption:
  - Q_{joint} can be additively decomposed

$$R(s, \boldsymbol{a}) = \sum_{i=1}^{N} r^i(\tau^i, a^i)$$

$$Q_{joint}(s, \boldsymbol{a}) \approx \sum_{i=1}^{N} Q^i(\tau^i, a^i)$$

Reasonable?



Value-Decomposition Network (VDN)

- Limitation of VDN
  - Expressive complexity of Q_{joint}:
    - Severely restricted by the decomposition design

$$Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{N} Q^i\left(\tau^i, a^i\right)$$

QMIX: Monotonic Value Function Factorisation

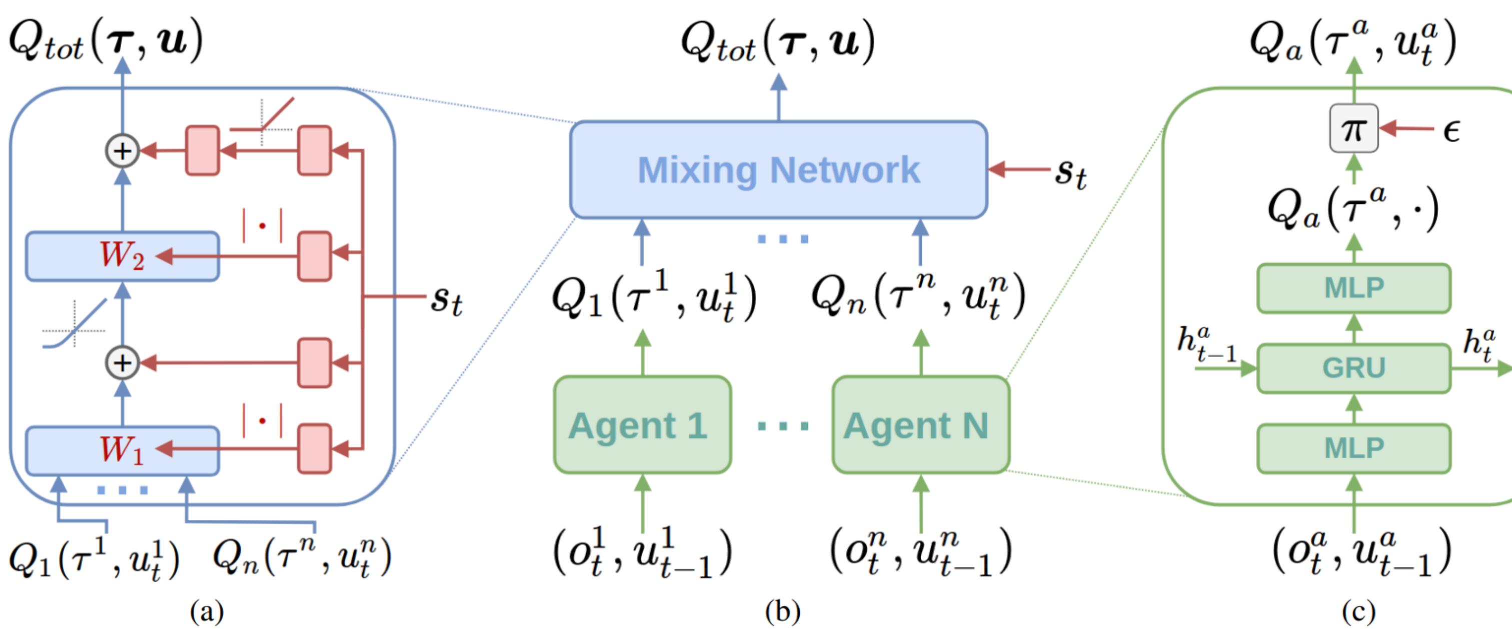
- QMIX:
  - "mixing network"
    - Learning weights and biases by neural networks:
      - Conditioned on global state s_t
      - Non-linear mixing
  - Satisfying monotonicity constraint:

$$\frac{\partial Q_{joint}(\boldsymbol{\tau}, \boldsymbol{u})}{\partial Q^i(\tau^i, u^i)} \geq 0, \forall i \in \mathcal{N}$$

  - Ensuring the monotonicity constraint by restricting the weights of the mixing network to be non-negative

QMIX: Monotonic Value Function Factorisation

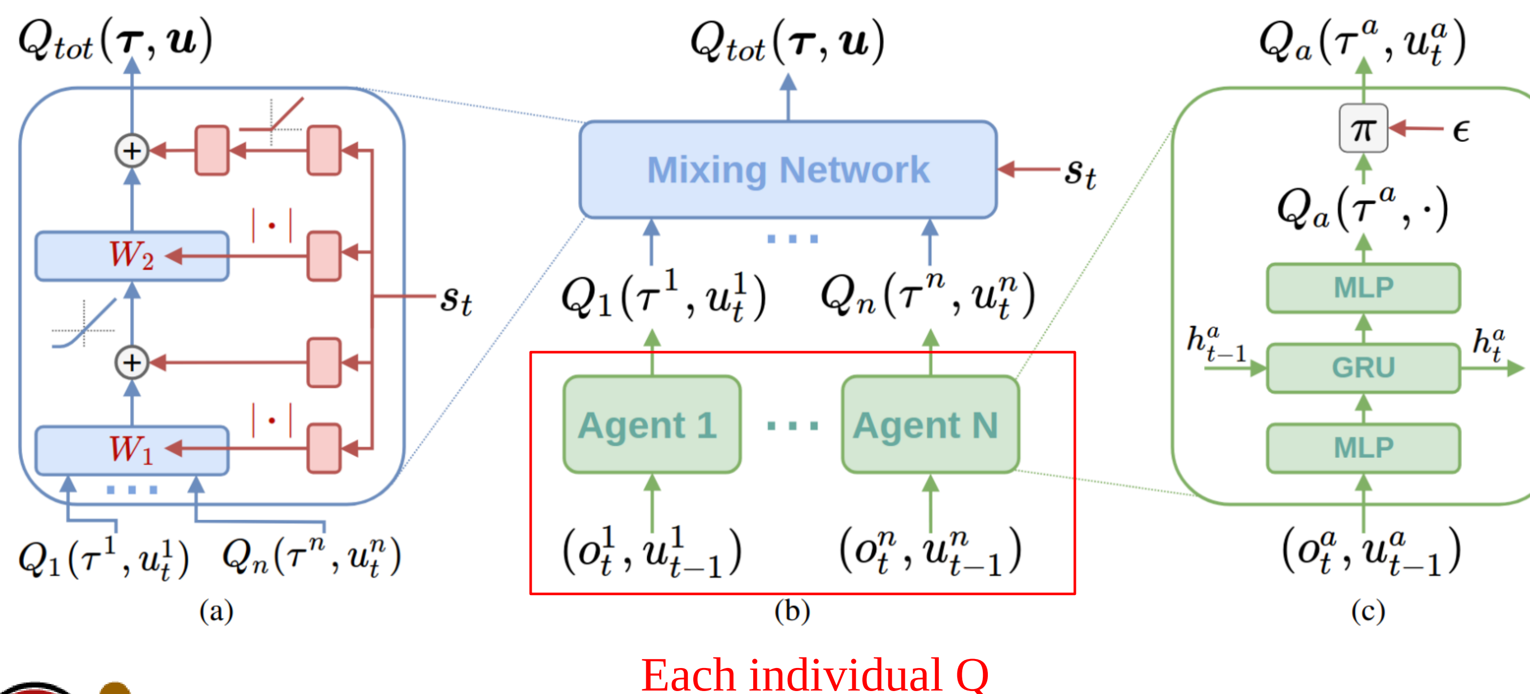
※ Q_{tot} equals to Q_{joint} here
※ Q_i equals to Q^i here



[QMIX]

QMIX: Monotonic Value Function Factorisation

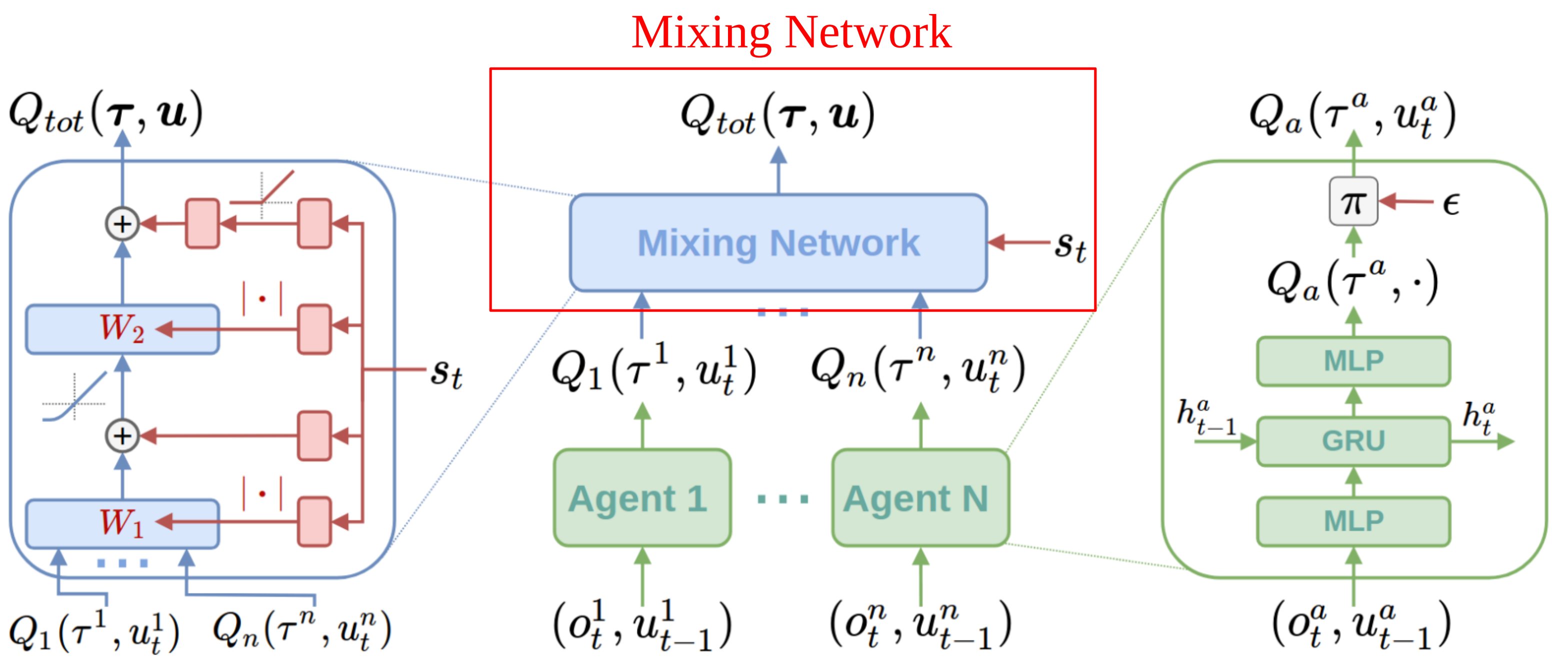
※ Q_{tot} equals to Q_{joint} here
※ Q_i equals to Q^i here



[QMIX]

QMIX: Monotonic Value Function Factorisation

※ Q_{tot} equals to Q_{joint} here
※ Q_i equals to Q^i here



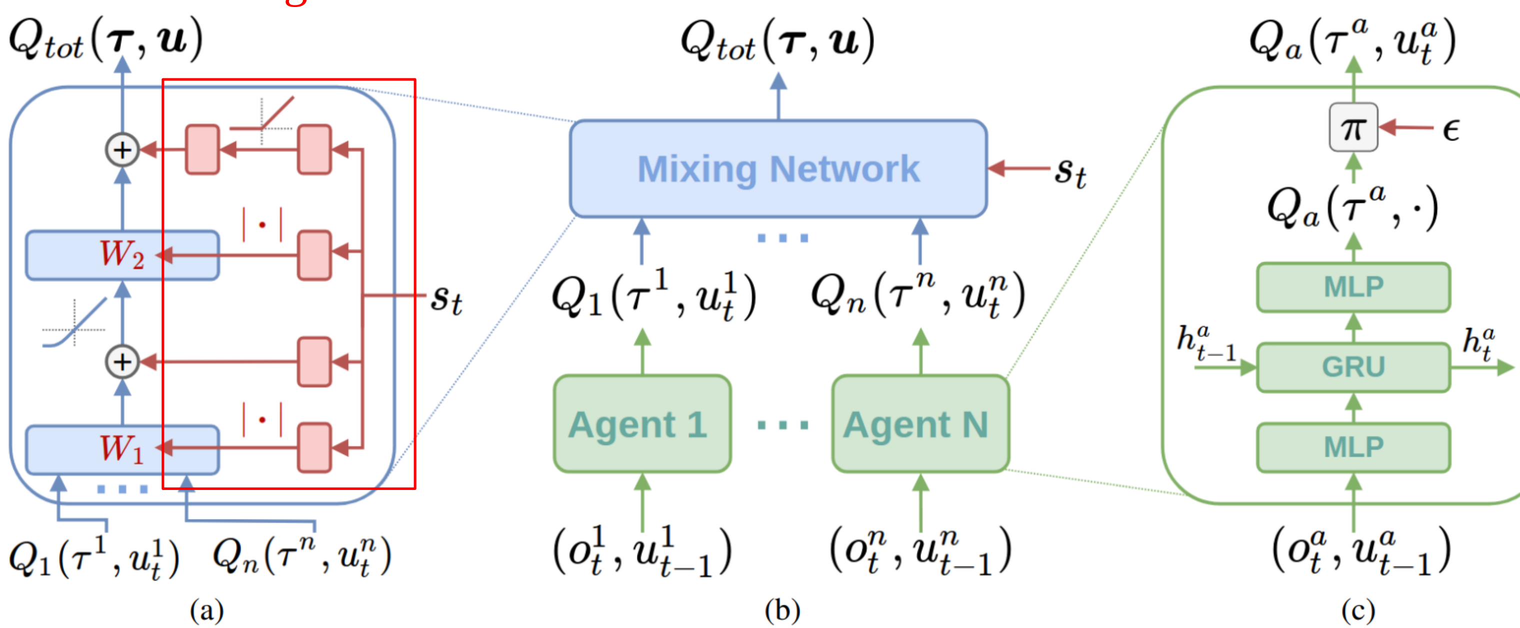
(a) (b) (c)

[QMIX]

QMIX: Monotonic Value Function Factorisation

※ Q_{tot} equals to Q_{joint} here
※ Q_i equals to Q^i here

Weights and biases are conditioned on state *s*

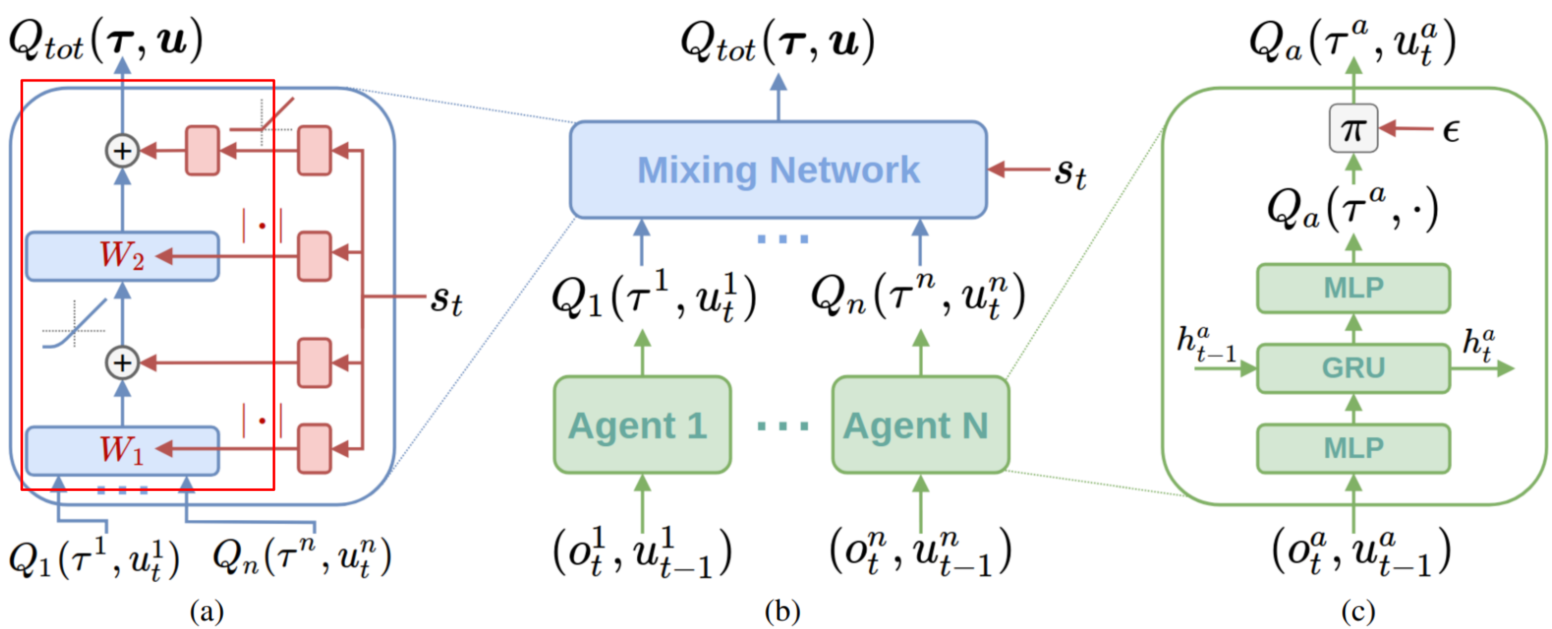


[QMIX]

QMIX: Monotonic Value Function Factorisation

※ Q_{tot} equals to Q_{joint} here
※ Q_i equals to Q^i here

Multiple layers, non-linear mixing

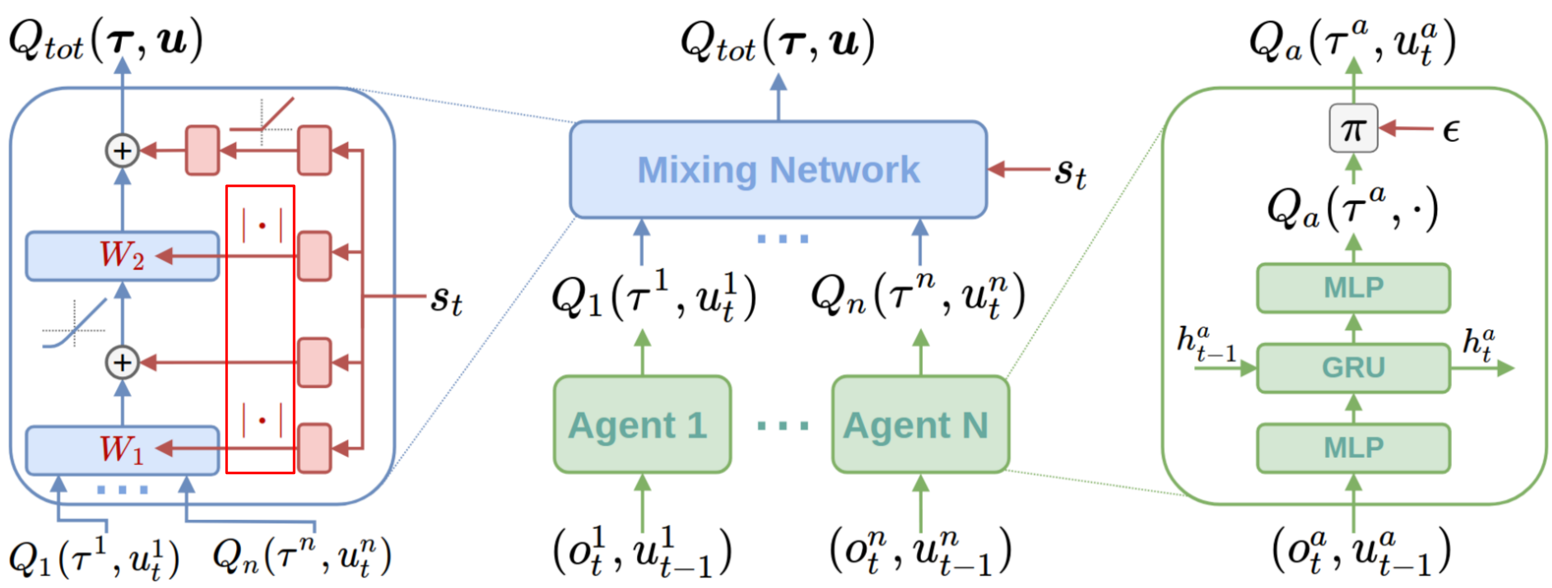


[QMIX]

QMIX: Monotonic Value Function Factorisation

※ Q_{tot} equals to Q_{joint} here
※ Q_i equals to Q^i here

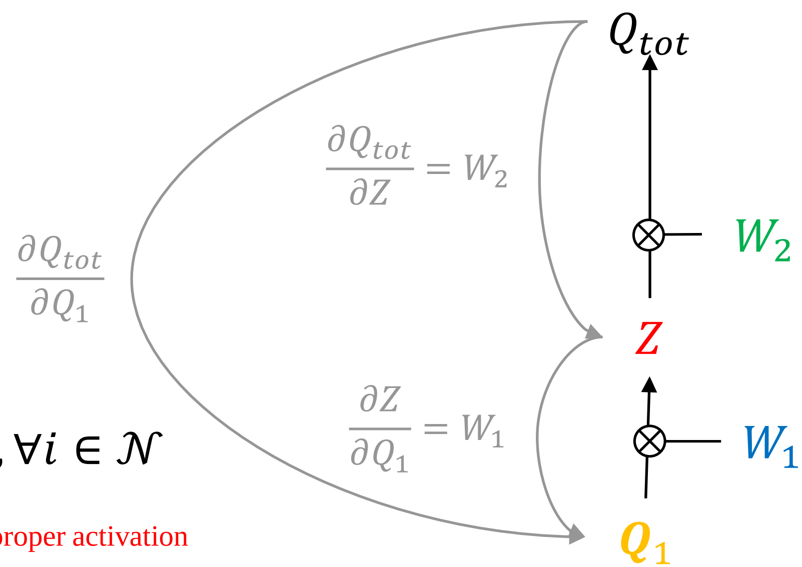
Ensure all weights are non-negative using **absolute** function

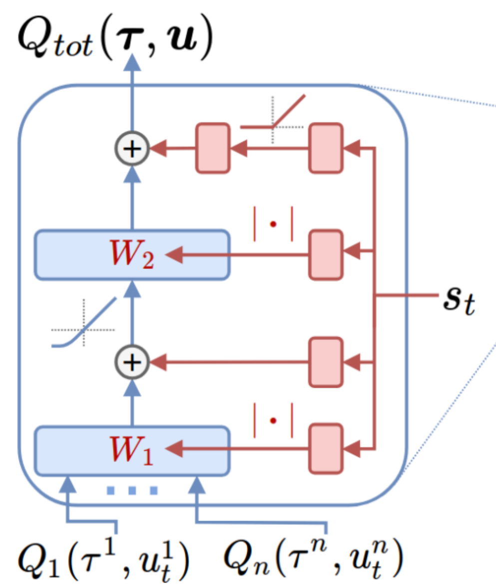


[QMIX]

Example

- Assume
  - 2 layers as the image shown
  - Assume embedding size = 1
  - Ignore biases and activation for simplicity



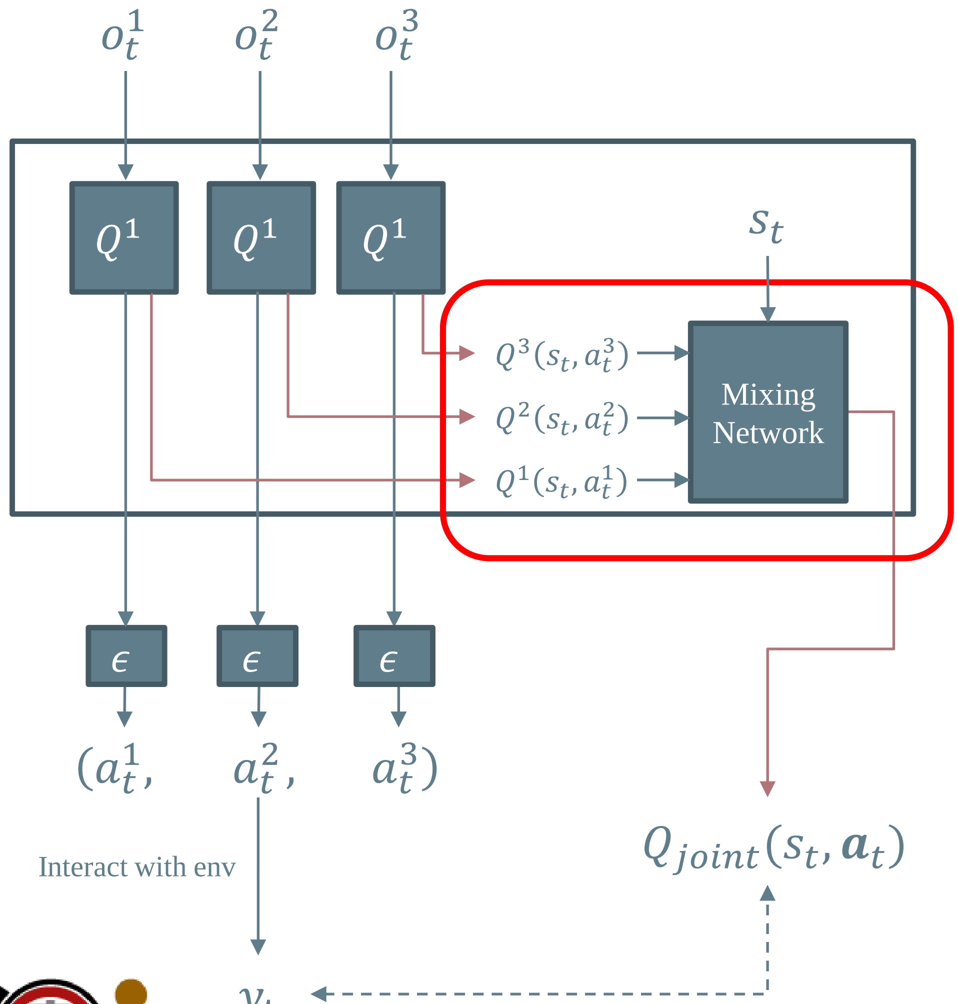


(a)

$$\frac{\partial Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q^i(\tau^i, a^i)} \geq 0, \forall i \in \mathcal{N}$$

With non-negative weights and proper activation

QMIX (CTDE Framework)



QMIX:

$$\frac{\partial Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q^i(\tau^i, a^i)} \geq 0, \forall i \in \mathcal{N}$$

IGM:

$$\begin{pmatrix} \arg\max_{a'} Q^1(\tau^1, a'), \\ \arg\max_{a'} Q^2(\tau^2, a'), \\ \arg\max_{a'} Q^3(\tau^3, a'), \end{pmatrix}$$

$$||$$

$$\arg\max_{\boldsymbol{a}'} Q_{joint}(\boldsymbol{\tau}, \boldsymbol{a}')$$

VDN vs QMIX: Expressive Complexity

VDN	$Q_{joint}(\boldsymbol{\tau},\boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i,u^i)$	Satisfy →	IGM $\arg\max_{\boldsymbol{a}} Q_{joint}(\boldsymbol{\tau},\boldsymbol{a}) = \begin{pmatrix} \arg\max_{a_1} Q^1(\tau^1,a^1) \\ \arg\max_{a_2} Q^2(\tau^2,a^2) \\ \dots \\ \arg\max_{a_N} Q^N(\tau^N,a^N) \end{pmatrix}$
QMIX	$\frac{\partial Q_{joint}(\boldsymbol{\tau},\boldsymbol{u})}{\partial Q^i(\tau^i,u^i)} \geq 0, \forall i \in \mathcal{N}$	Satisfy →	IGM $\arg\max_{\boldsymbol{a}} Q_{joint}(\boldsymbol{\tau},\boldsymbol{a}) = \begin{pmatrix} \arg\max_{a_1} Q^1(\tau^1,a^1) \\ \arg\max_{a_2} Q^2(\tau^2,a^2) \\ \dots \\ \arg\max_{a_N} Q^N(\tau^N,a^N) \end{pmatrix}$
VDN	$Q_{joint}(\boldsymbol{\tau},\boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i,u^i)$	Satisfy →	QMIX $\frac{\partial Q_{joint}(\boldsymbol{\tau},\boldsymbol{u})}{\partial Q^i(\tau^i,u^i)} \geq 0, \forall i \in \mathcal{N}$

Both satisfy IGM, but QMIX restricts less
→ **QMIX can represent a much richer class of $\boldsymbol{Q_{joint}}$**

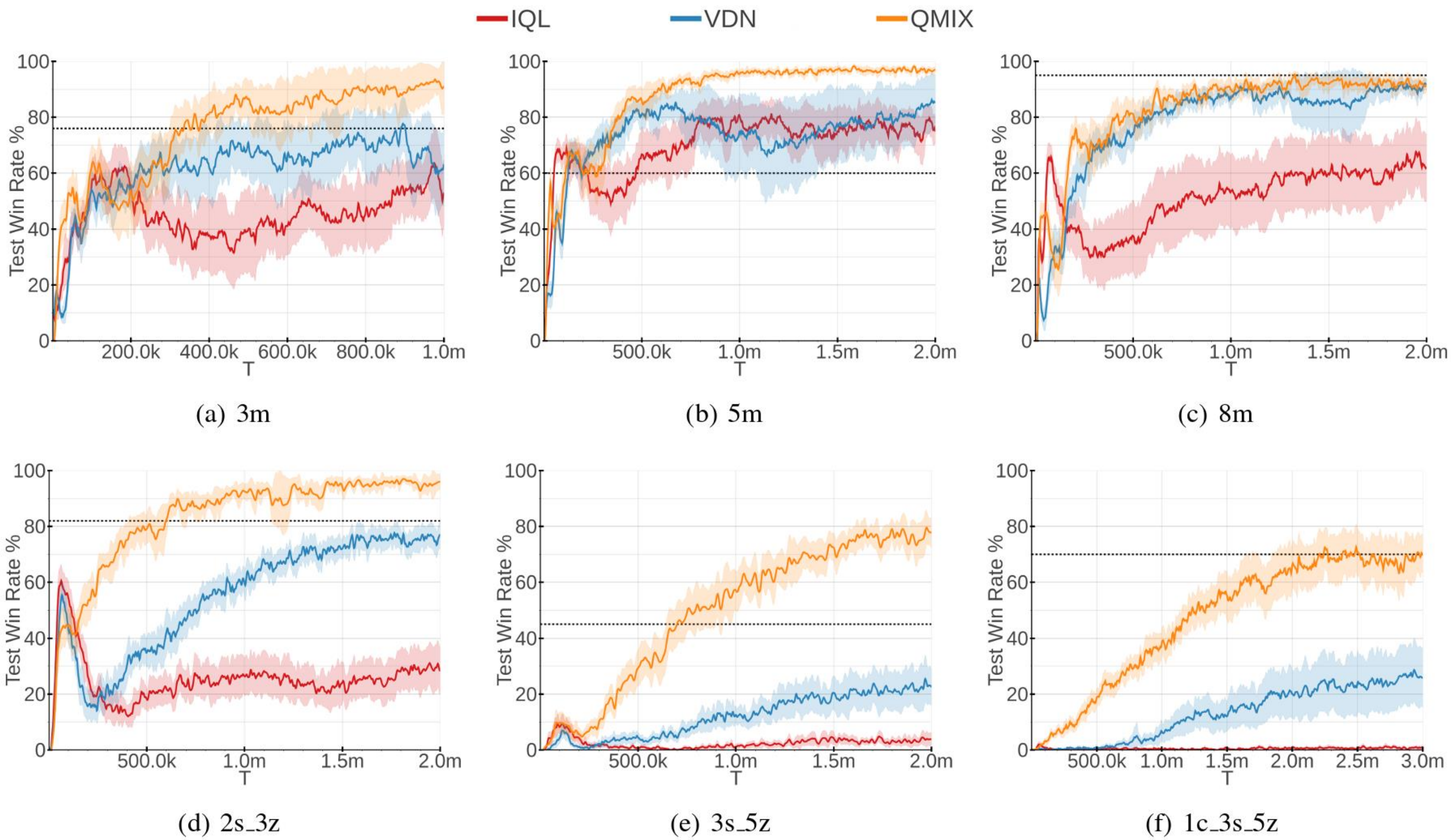
Experiment

- Decentralized StarCraft Micromanagement
  - Control individual units' positioning and attack commands as they fight enemies
  - Each unit is controlled by a decentralized controller
- Scenarios with symmetric teams
  - 3 marines (3m)
  - 5 marines (5m)
  - 8 marines (8m)
  - 2 stalkers with 3 zealots (2d 3z)
  - 3 stalkers and 5 zealots (3s 5z)
  - 1 colossus, 3 stalkers and 5 zealots (1c 3s 5z)

(a) 5 Marines map

(b) 2 Stalkers & 3 Zealots map

Experiment Results



(a) 3m (b) 5m (c) 8m

(d) 2s_3z (e) 3s_5z (f) 1c_3s_5z

Decentralized StarCraft Micromanagement

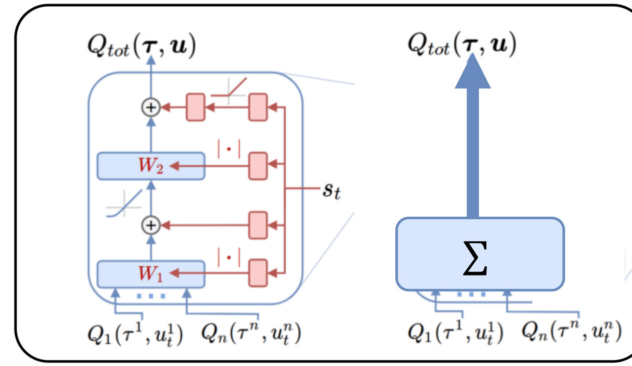
※ Dotted line: heuristic method

Ablation Study

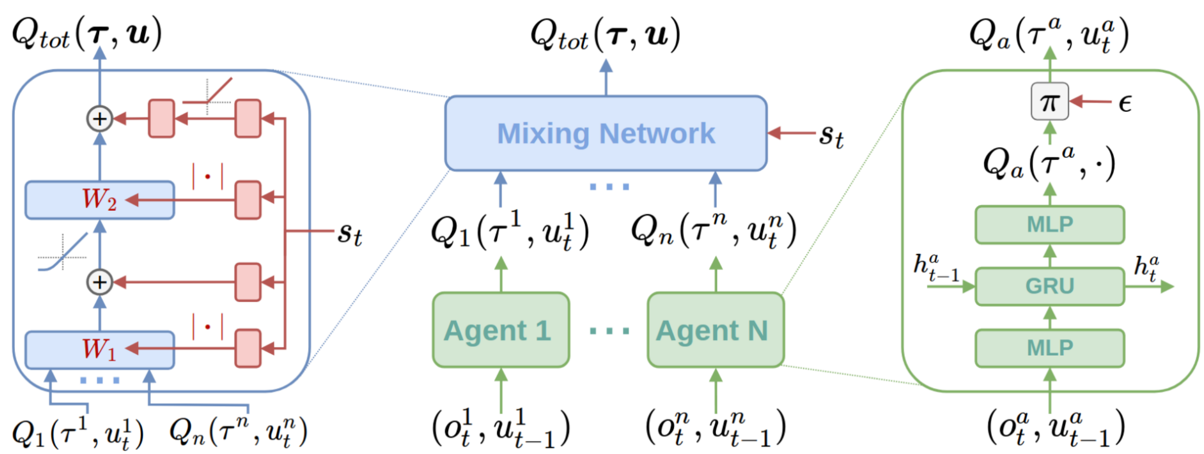
- VDN

$$Q_{joint}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i, u^i)$$

- QMIX



QMIX VDN



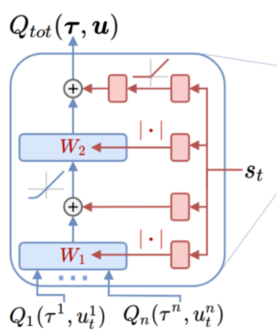
(a) (b) (c)

Ablation Study

- VDN-S
  - It is an extension VDN by adding a state-dependent term to the sum of the agent's Q-Values

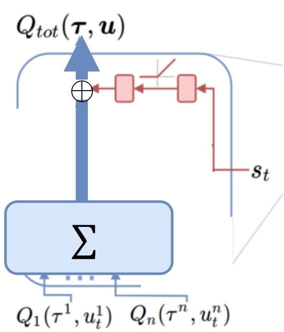
$$Q_{joint}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} Q^i(\tau^i, u^i) \quad +bias(s_t)$$

  - Goal: investigate the significance of utilizing the state s in comparison to the non-linear mixing



QMIX

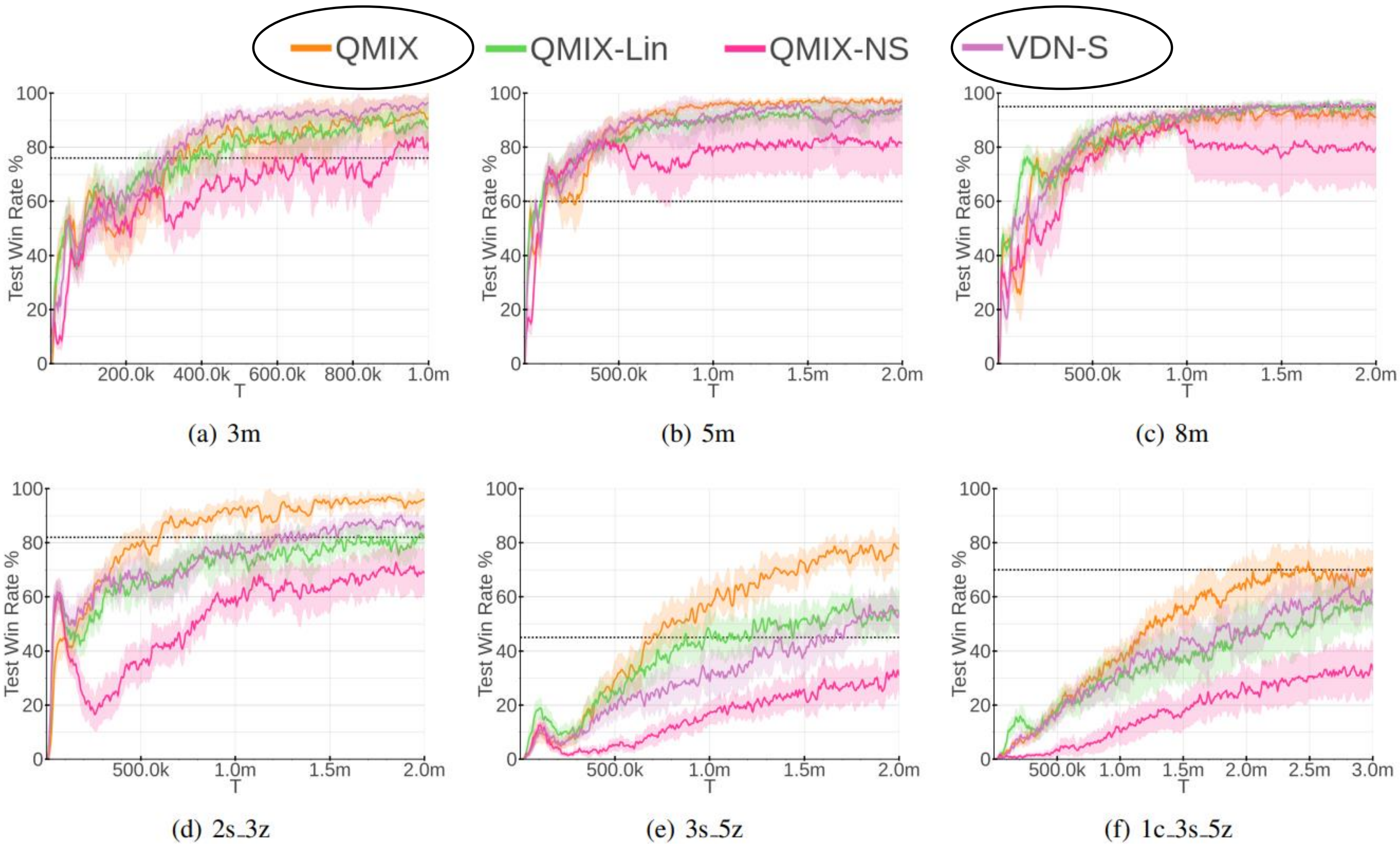
VS



VDN-S

I-Chen Wu

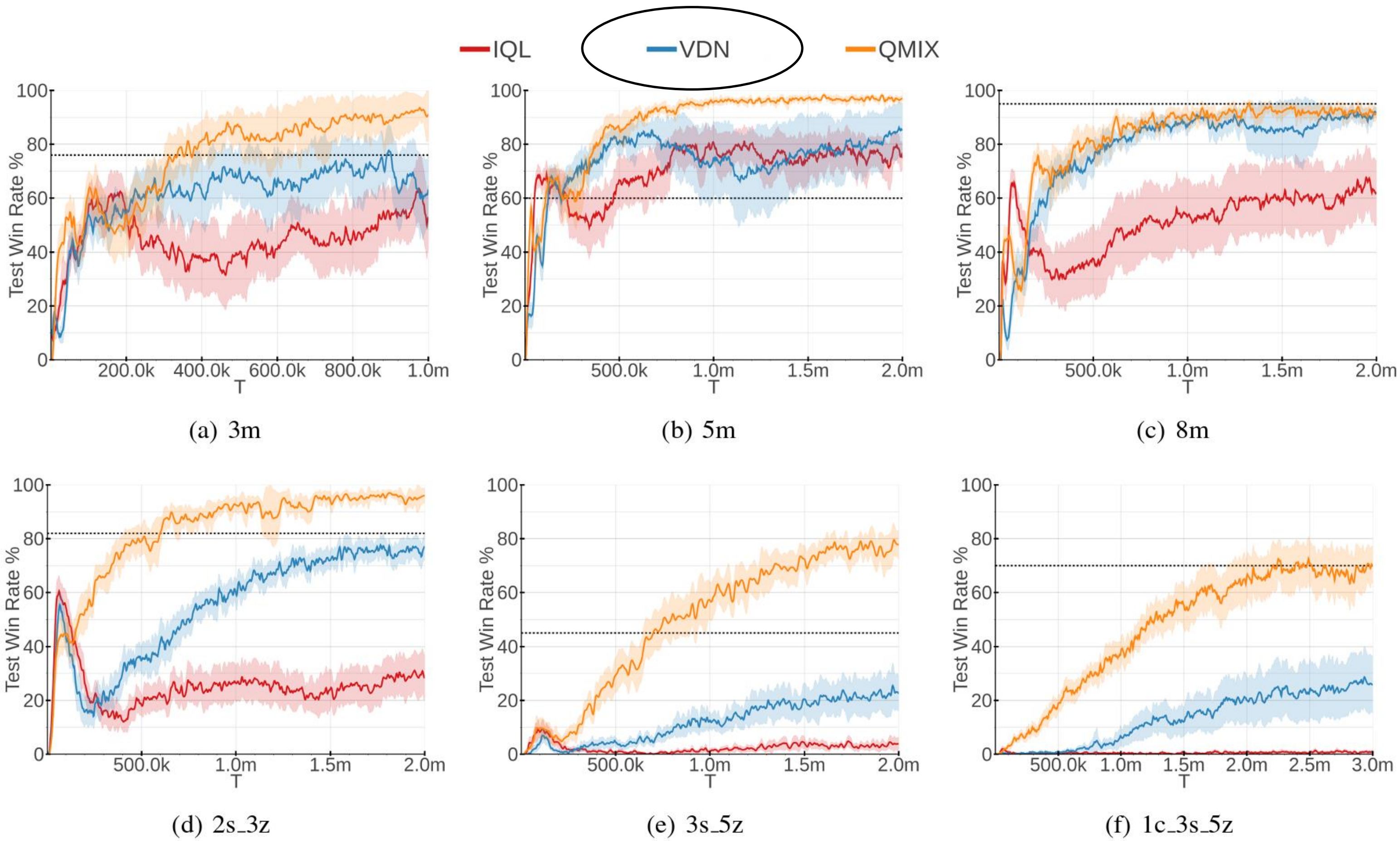
Ablation Study



(a) 3m (b) 5m (c) 8m

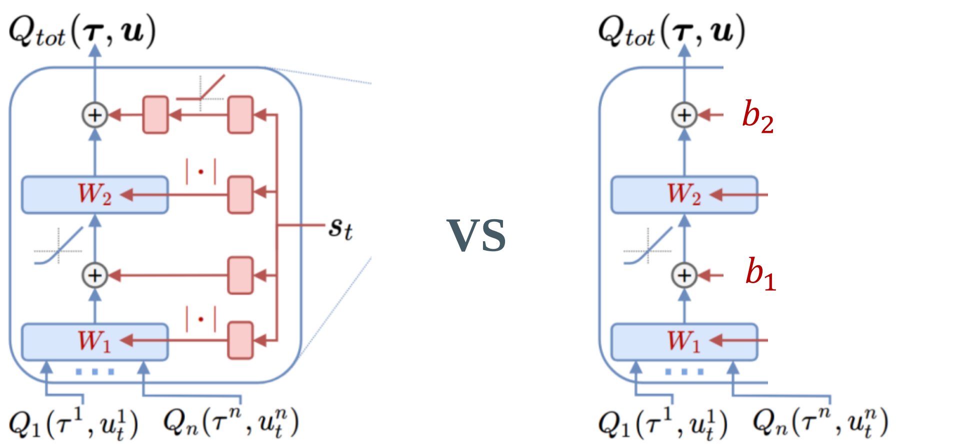
(d) 2s_3z (e) 3s_5z (f) 1c_3s_5z

Ablation Study



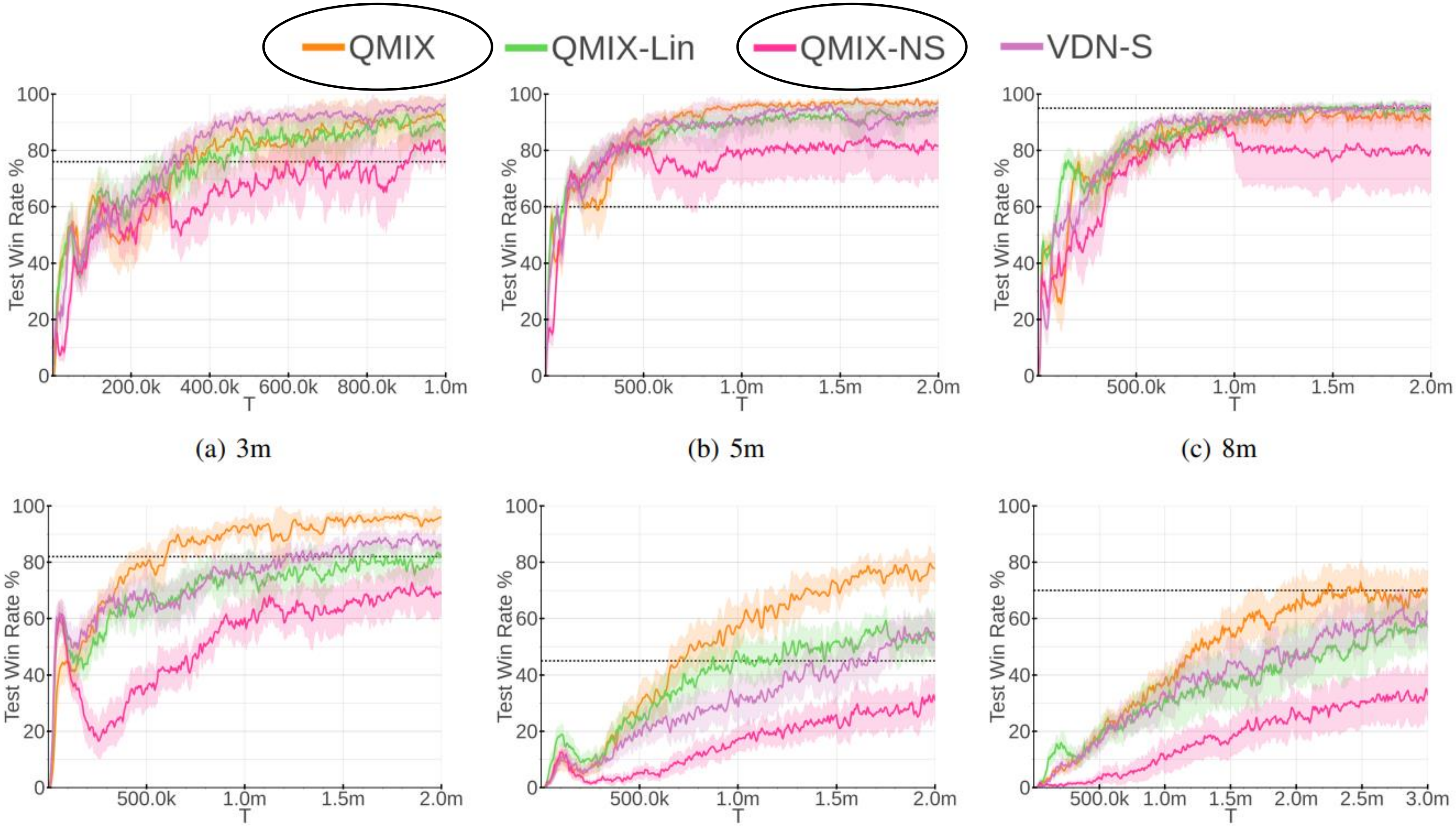
Ablation Study

- QMIX-NS
  - The weights and biases of the mixing network are learned in the standard way, without conditioning on the state and without hypernetworks
  - Goal: analyze the significance of extra state information on the mixing network



QMIX QMIX-NS

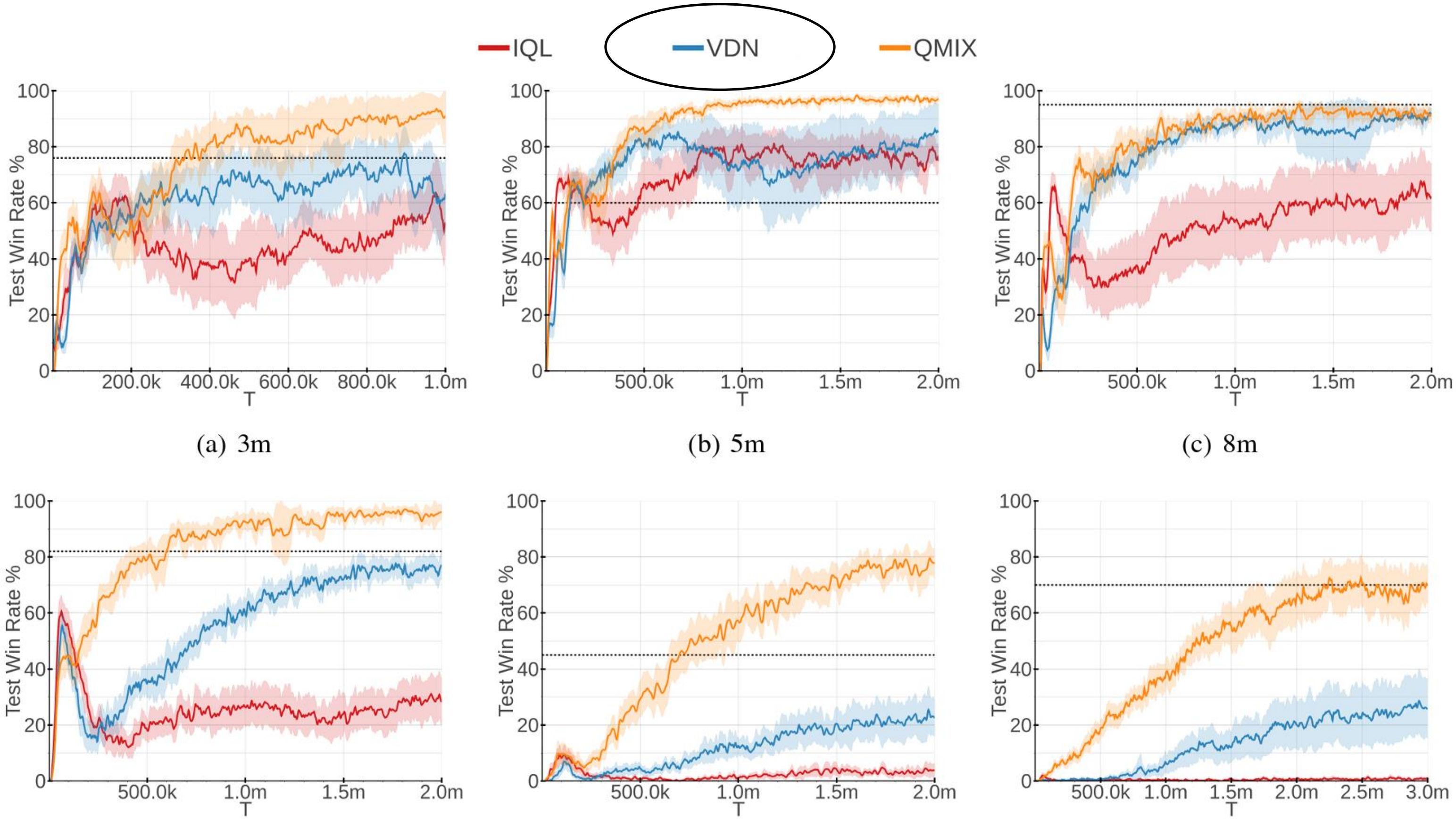
Ablation Study



(a) 3m (b) 5m (c) 8m

(d) 2s_3z (e) 3s_5z (f) 1c_3s_5z

Ablation Study

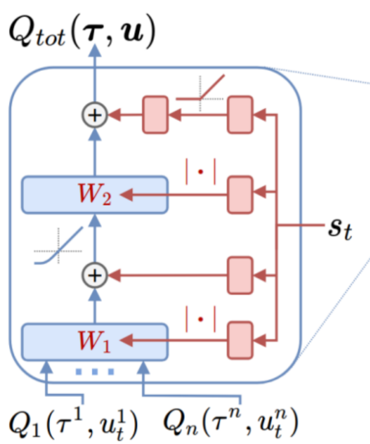


(a) 3m (b) 5m (c) 8m

(d) 2s_3z (e) 3s_5z (f) 1c_3s_5z

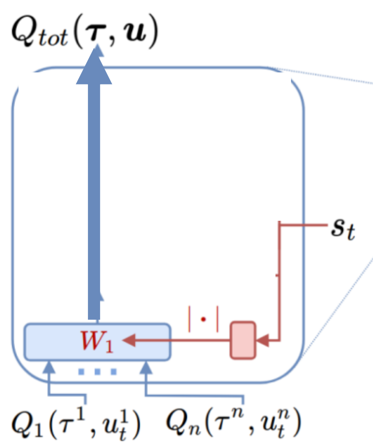
Ablation Study

- QMIX-Lin
  - An extension of VDN that uses the state s to perform a weighted sum over Q^i values
  - Goal: investigate the necessity of non-linear mixing by removing the hidden layer of the mixing network



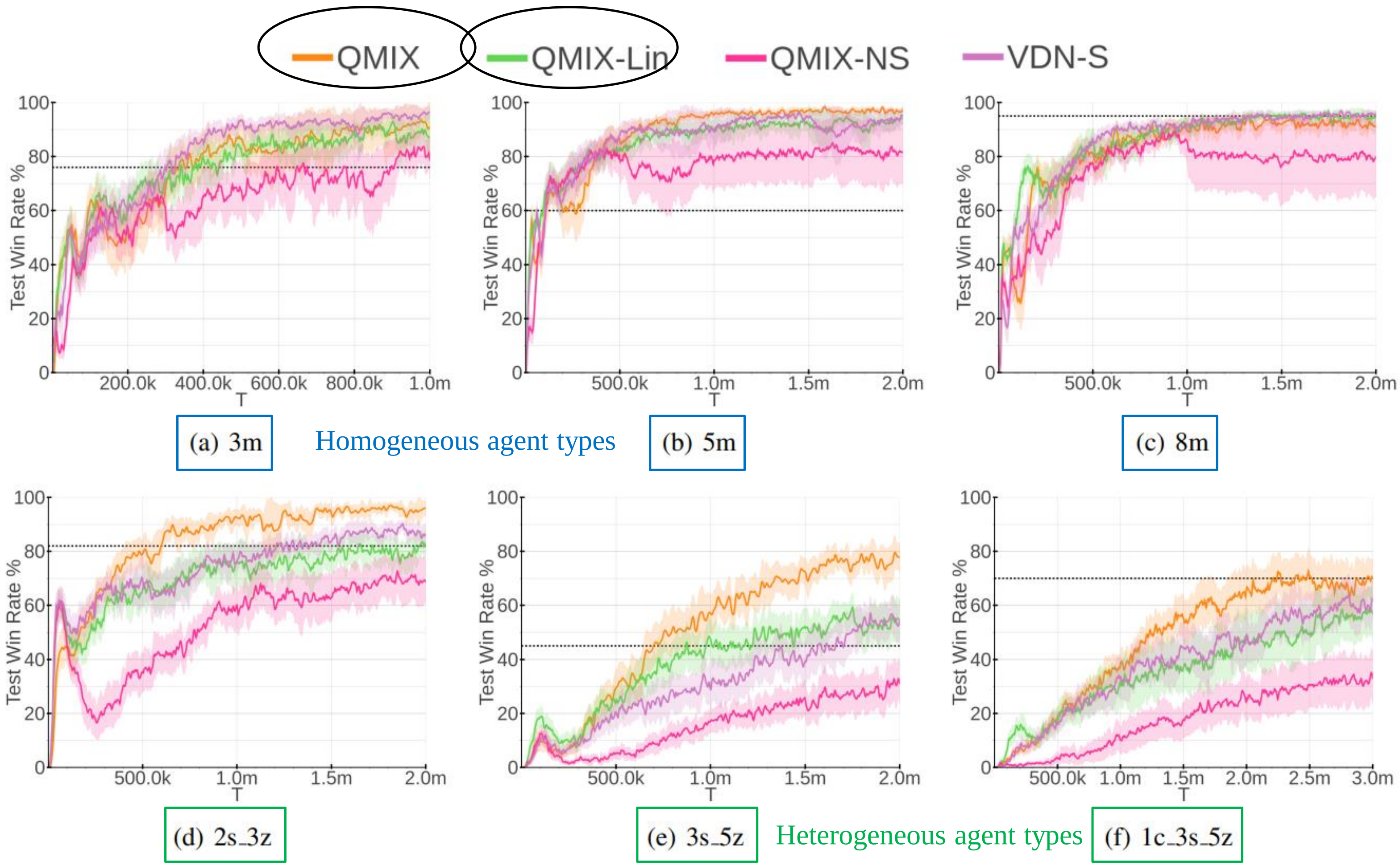
QMIX

VS



QMIX-Lin

Ablation Study



- Non-linear factorization is not always required
- Required on maps with heterogeneous agent types

Summary

- Cooperative MARL is introduced today
  - Categories of MARL
  - Problem formulation of Cooperative MARL
- Value-based Cooperative MARL methods
  - CTDE framework
  - VDN:
    - Value factorization: sum up all
  - QMIX:
    - Value factorization:
      - Learn weights using a network
      - Non-linear mixing
      - Conditioned on state