Interpretable and Efficient Multi-Agent Reinforcement Learning

Presented by: **Zichuan Liu**

zichuanliu@smail.nju.edu.cn

Nanjing University

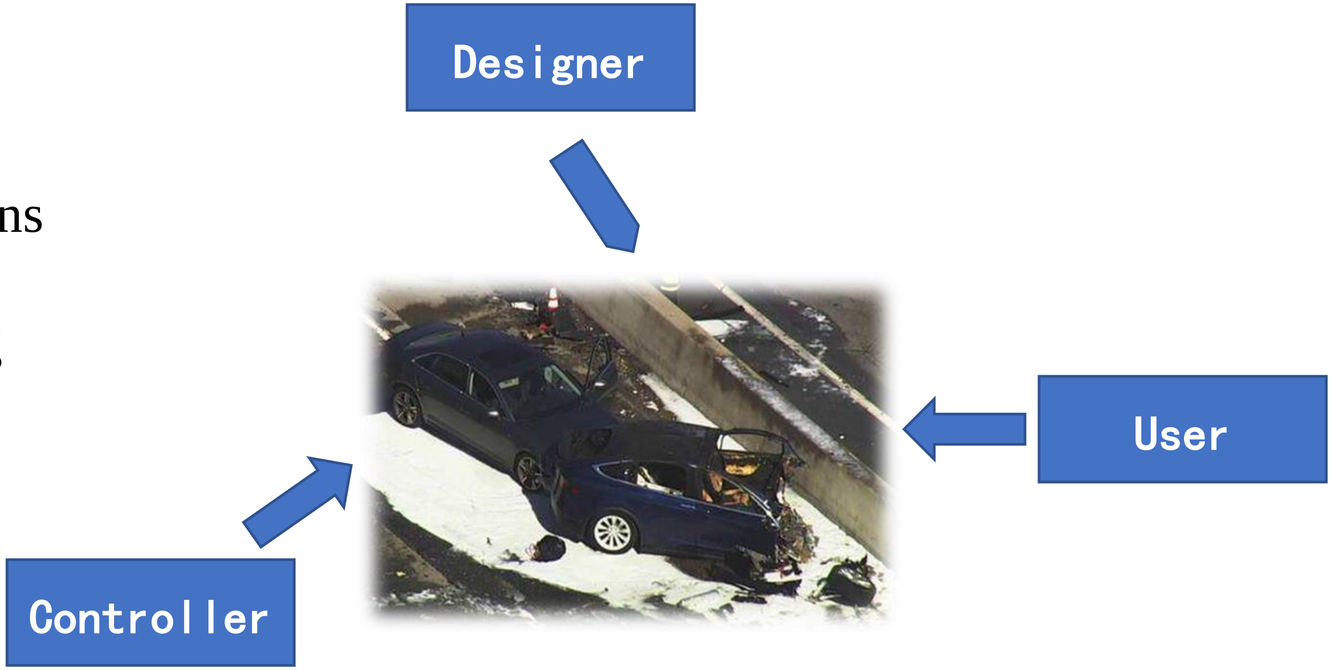
Content

- **White-Boxes Modelling Decision-Making**
- **Intrinsic Explanation of Credit Assignment**
- **Efficient Sample and Action Space Pruning**

Background

Black-box models with post-hoc explanation techniques might not suffice in MARL:

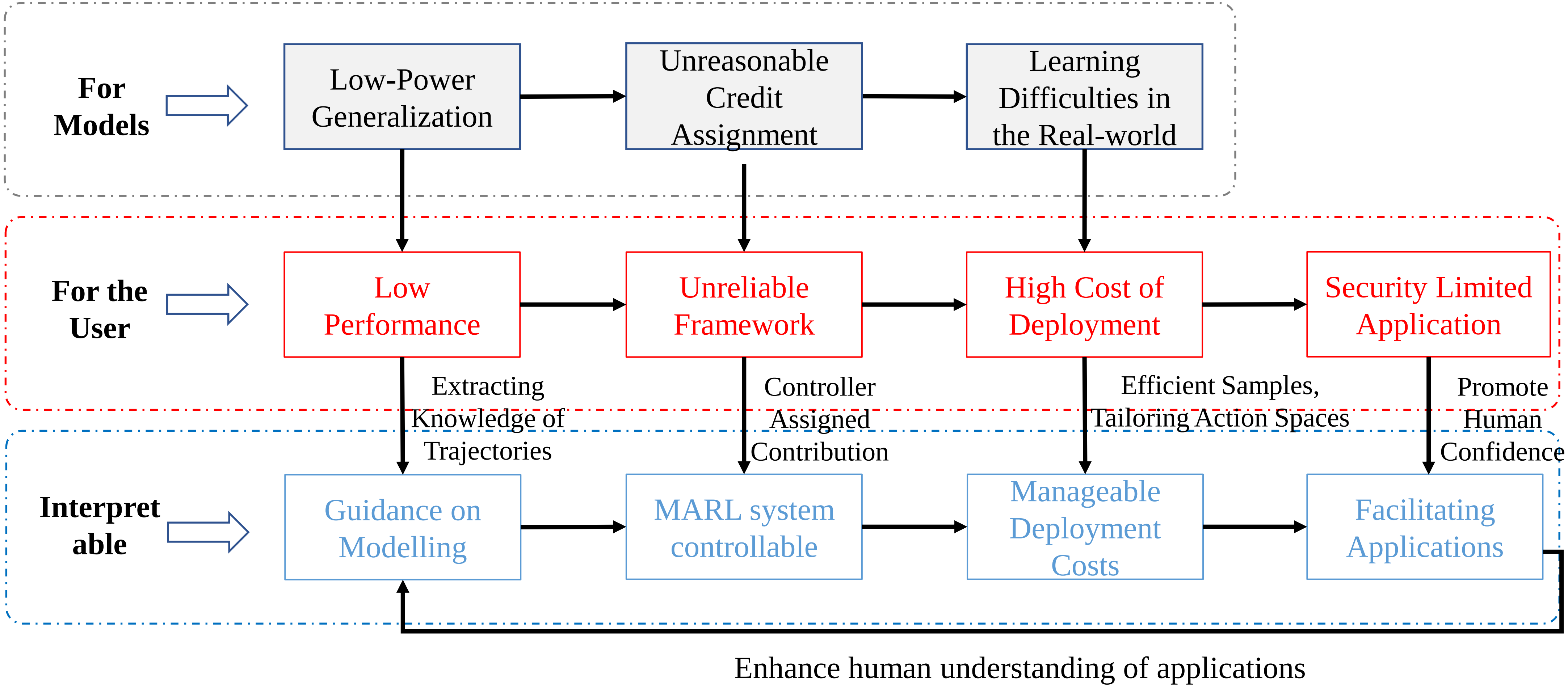
- Lack of transparency in credit assignment
- Misrepresent agents’ decisions
- Expensive and unstable calculations
- Low sample efficiency in learning
- The action space is too large
- … …



How do we explain multiple Tesla autonomous driving accidents?

Background

Why Interpretable and Efficient MARL?



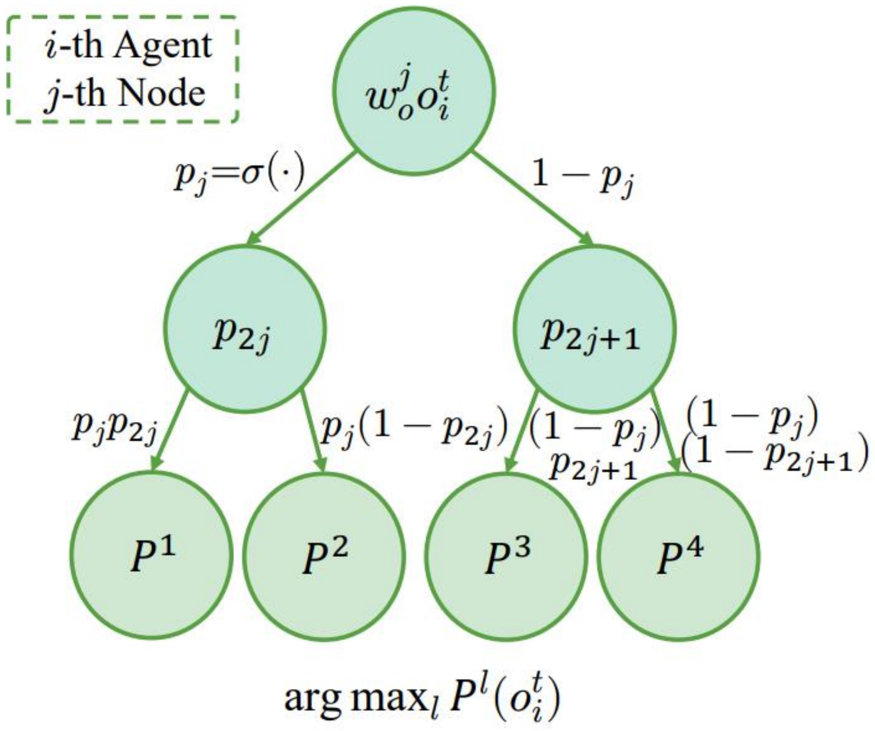
MIXRTs: Toward Interpretable Multi-Agent Reinforcement Learning via Mixing Recurrent Soft Decision Trees

Zichuan Liu, Yuanyang Zhu, Zhi Wang, Yang Gao, Chunlin Chen

南京大學
NANJING UNIVERSITY

Motivation

- Approximator for Q-learning via Soft Decision Trees (SDTs)
  - Soft decision boundary with gradient descent
  - Decision structure is simple and expressive
  - With the inherent feature interpretation
- Through a recursive structure to capture the history of agents
- Make the paradigm of CTDE an interpretable central controller



A two-level Soft Decision Tree

Methodology

- Recurrent Tree Cells, RTCs
  - each non-leaf node traverses to its left child node

$$p_j(o_i^t, h_i^{t-1}) = \sigma(w_o^j o_i^t + w_h^j h_i^{t-1} + b^j)$$

  - The path probability of the top node to leaves

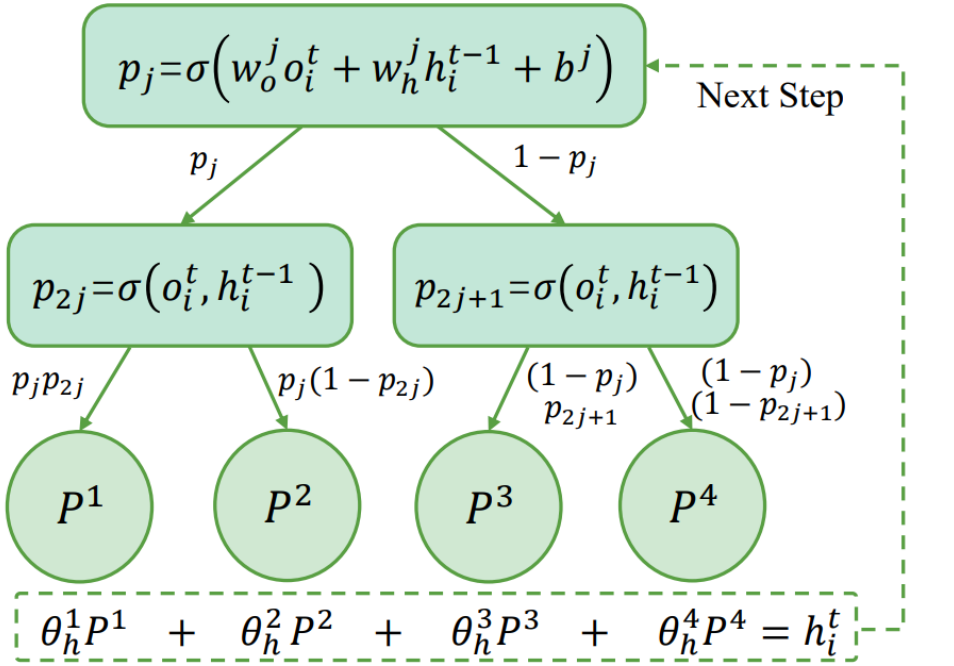
$$P^l(o_i^t, h_i^{t-1}) = \prod_{j \in route(l)} p_{\lfloor j/2 \rfloor \to j}(o_i^t, h_i^{t-1})^{[l \swarrow j]} \left(1 - p_{\lfloor j/2 \rfloor \to j}(o_i^t, h_i^{t-1})\right)^{1-[l \swarrow j]}$$

  - Obtain the current state

$$h_i^t = \sum_{l \in LeafNodes} P^l(o_i^t, h_i^{t-1})\, \theta_h^l$$

  - Obtain the action-observation value

$$Q_i(\tau_i, \cdot) = h_i^t w_q$$



The process of a two-level Recurrent Tree Cell

Methodology

- Ensemblling multiple RTCs with Low Variance
  - Individual trees vary widely

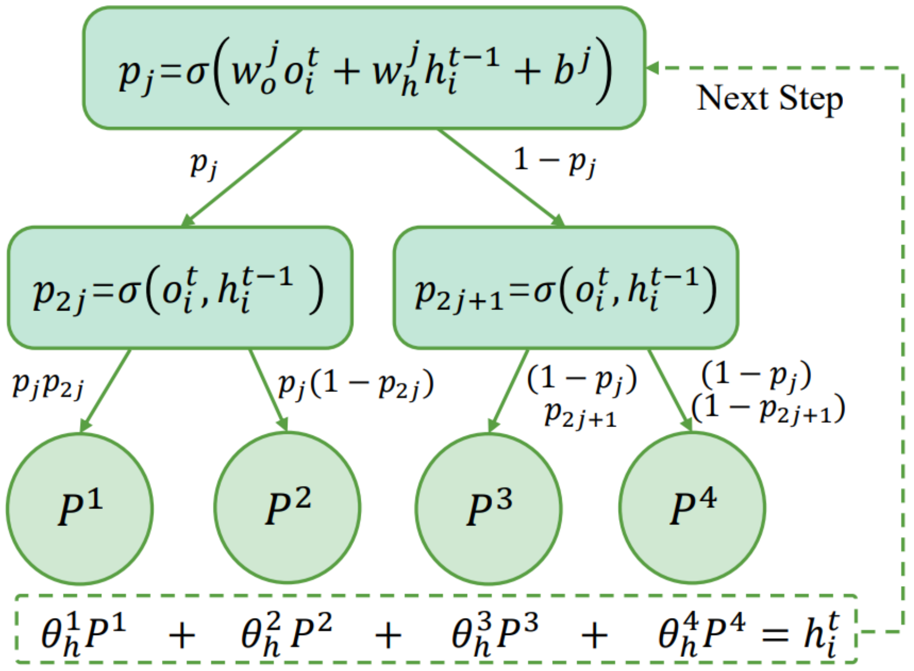
$$Q_i(\tau_i, \cdot) = h_i^t w_q$$

  - RTCs utilize a linear ensemble

$$Q_i(\tau_i, \cdot) = h_{i,(1)}^t w_{q,(1)} + h_{i,(2)}^t w_{q,(2)} + \ldots + h_{i,(H)}^t w_{q,(H)}$$

  - Rewritten the current state

$$h_i^t = \left[h_{i,(1)}^t, h_{i,(2)}^t, \cdots, h_{i,(H)}^t\right]$$



The process of a two-level Recurrent Tree Cell

Methodology

- The mixing tree decomposes the joint value into individual values
  - Individual action-values

$$Q_i(\tau_i,\cdot) = h^t_{i,(1)}\, w_{q,(1)} + h^t_{i,(2)}\, w_{q,(2)} + \ldots + h^t_{i,(H)}\, w_{q,(H)}$$

  - The structure of the mixing tree
    - Input individual values and the global state

$$p_j(Q_i, s^t) = \sigma(w_q^j Q_i + w_s^j s^t + b^j)$$

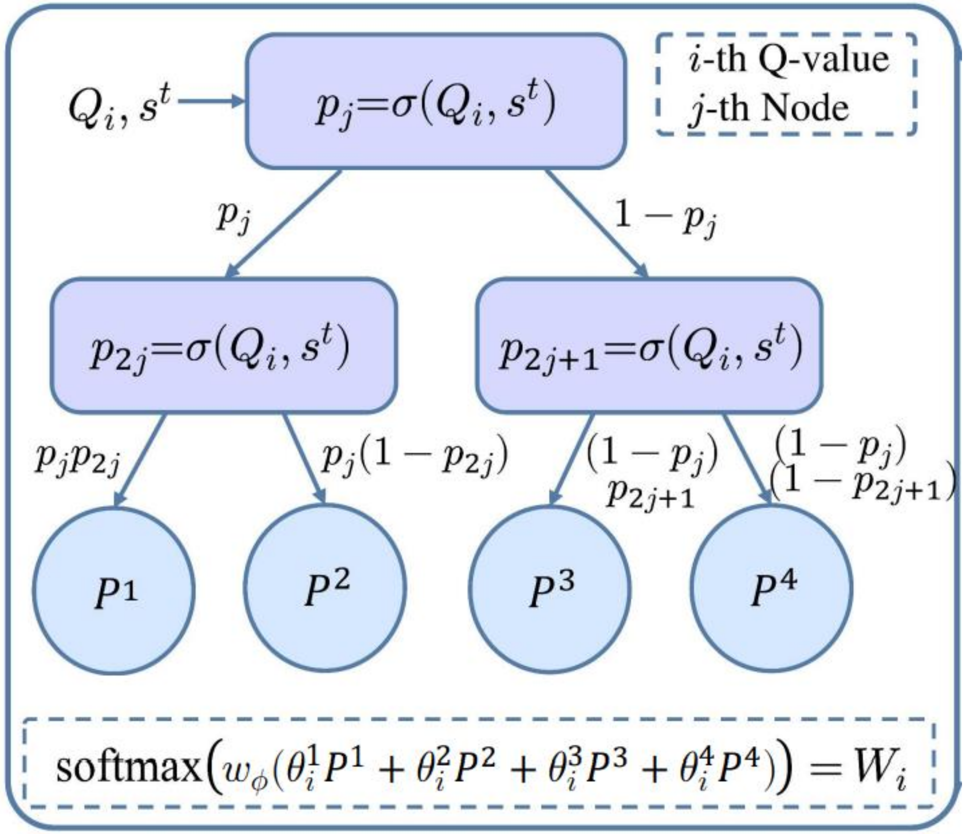
    - weighted in a linear manner

$$\phi_i = \sum_{l \in LeafNodes} P^l(Q_i, s^t)\, \theta_i^l,$$

$$W_i = \frac{exp(\sum_{k=1}^{H} \phi_{i,(k)}\, w_{\phi,(k)})}{\sum_{i=1}^{n} exp(\sum_{k=1}^{H} \phi_{i,(k)}\, w_{\phi,(k)})},$$

    - Obtain the joint action-observation value

$$Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}) \approx \sum_{i=1}^{n} W_i Q_i(\tau_i, u_i)$$



The structure of the mixing tree

Overall Architecture

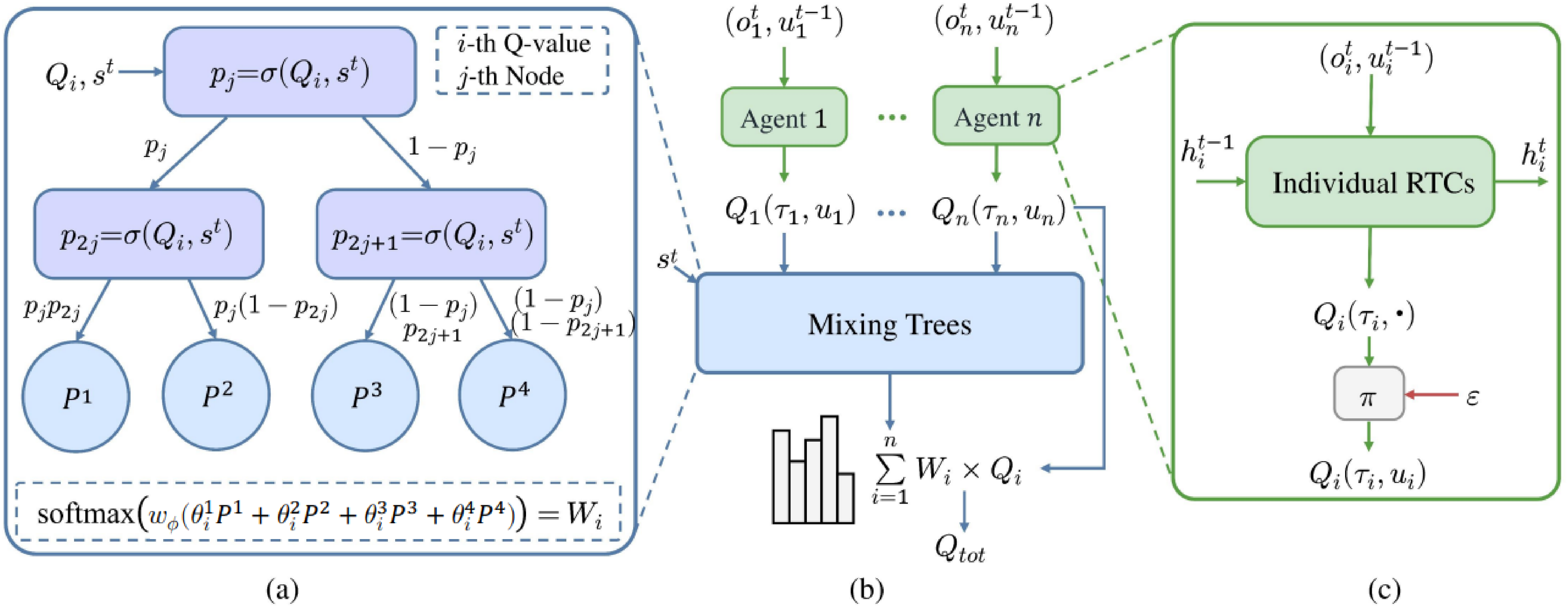
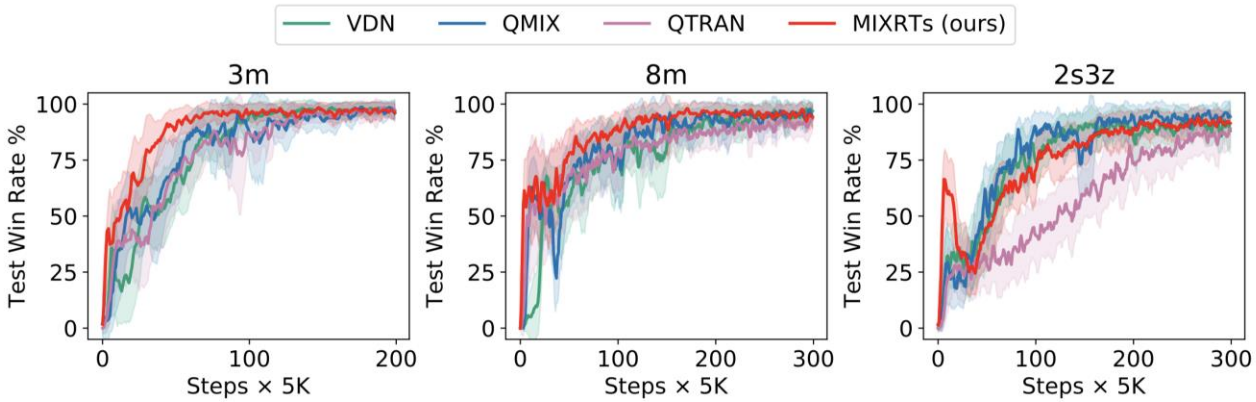


Fig. 2. MIXRTs architecture. (a) Diagram of the structure of the mixing tree with depth 2. (b) In the overall MIXRTs architecture, we finally obtain the joint Q_{tot} value via a linear combination of the individual action-value functions. (c) Individual RTCs for each agent.

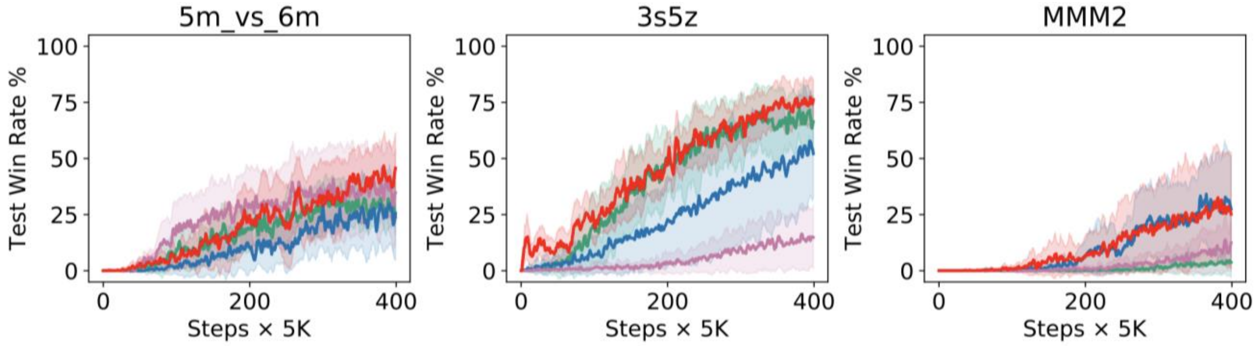
Experiments

- Performance on simple scenarios
    - Fast mastery of simple tasks
    - Models achieve high win rates



Performance on simple scenarios

- Performance on difficult scenarios
    - Achieve competitive performance
    - More stability during learning



Performance on hard scenarios

- Comparison of model parameters
    - Linear model, fewer parameters

Method	3m	2s3z	5m_vs_6m	3s5z	8m_vs_9m	MMM2	6h_vs_8z
VDN	28, 297	31, 883	30, 412	35, 534	32, 911	39, 250	32, 206
QMIX	37, 738	62, 892	55, 789	111, 951	96, 304	173, 651	72, 847
QTRAN	70, 911	84, 437	80, 518	101, 492	94, 513	120, 320	88, 436
MIXRTs (ours)	20, 880	34, 448	28, 752	48, 560	38, 592	62, 736	35, 440

Inherent Interpretation

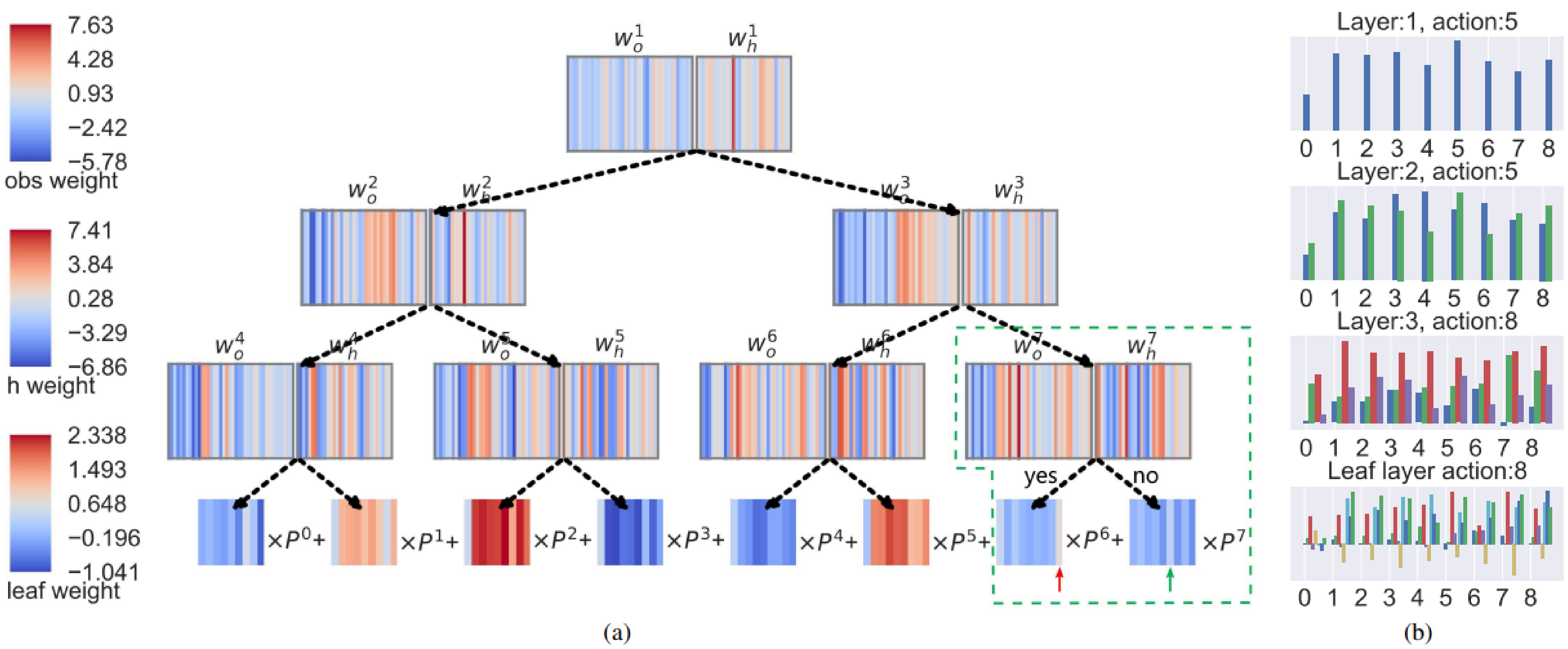
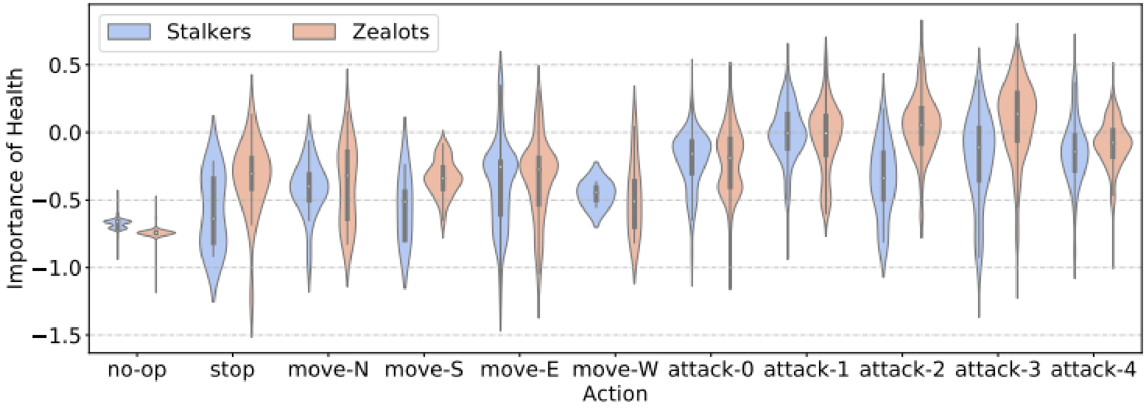


Fig. 7. Heatmap visualization of earned filters and action distributions for each layer. (a) Heatmap visualization of the learned filters in the learned RTCs of depth 3. The weights of each non-leaf node feature contain the current observations (left) and the historical records (right), respectively. The leaf nodes indicate the magnitude of the different action distributions. (b) Actions probability distribution of the nodes of the RTCs with a given observation, where each node is distinguished by a different color bar.

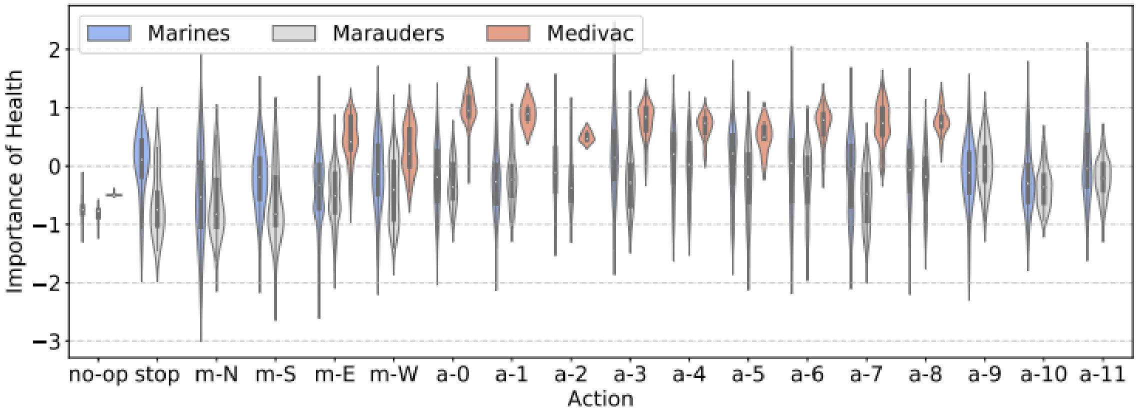
Stability analysis

Perturbation tests the win rate of MIXRTs. To quantify the interpretation, we use the trained model to calculate the important features in each step via Eq. (13). Then, we mask a varying percentage of the least and most important features with zeros for each step and redo the decision-making.

Least important data	3m	2s3z	5m_vs_6m	Most important data	3m	2s3z	5m_vs_6m
Masking 0%	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$63.37_{\pm 2.10}$	Masking 0%	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$63.37_{\pm 2.10}$
Masking 5%	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$48.95_{\pm 1.72}$	Masking 5%	$83.48_{\pm 1.35}$	$93.87_{\pm 1.18}$	$13.37_{\pm 3.78}$
Masking 10%	$96.77_{\pm 1.74}$	$100.00_{\pm 0.00}$	$41.75_{\pm 2.57}$	Masking 10%	$63.78_{\pm 3.89}$	$41.19_{\pm 0.73}$	$0.00_{\pm 0.00}$
Masking 20%	$82.19_{\pm 1.46}$	$70.96_{\pm 5.26}$	$15.53_{\pm 3.24}$	Masking 20%	$12.90_{\pm 1.51}$	$3.22_{\pm 2.64}$	$0.00_{\pm 0.00}$
Masking 30%	$48.38_{\pm 4.15}$	$46.67_{\pm 3.19}$	$0.00_{\pm 0.00}$	Masking 30%	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$



(b) The violin plot of feature importance on 2s3z.



(c) The violin plot of feature importance on MMM2.

Case study

(a) 2s3z (step=10)

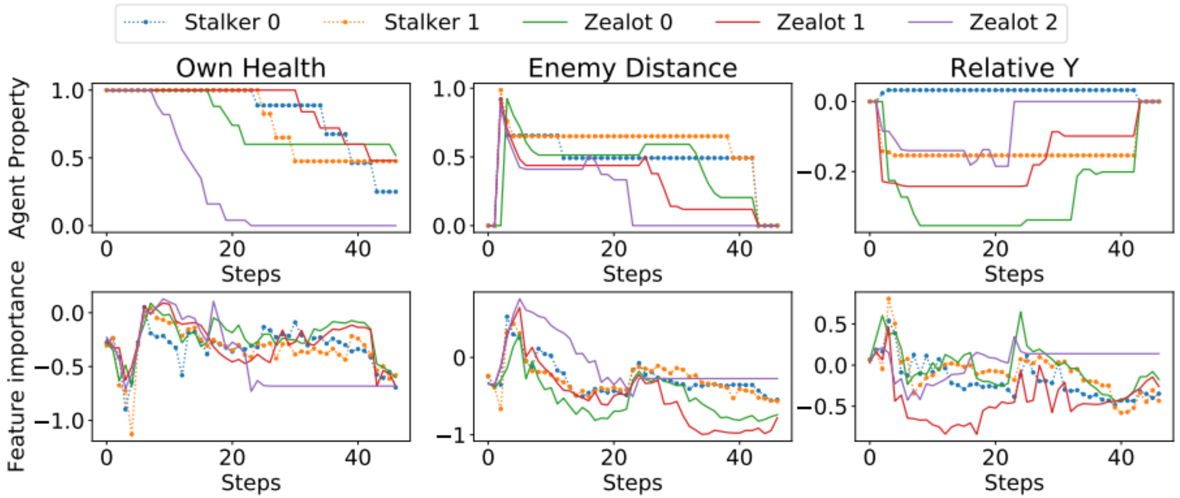
(b) 2s3z (step=27)

(c) 2s3z (step=34)

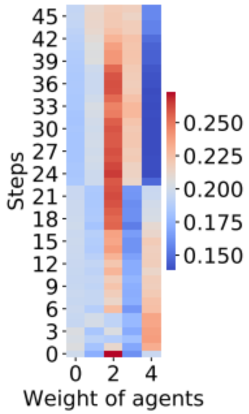
(d) MMM2 (step=7)

(e) MMM2 (step=12)

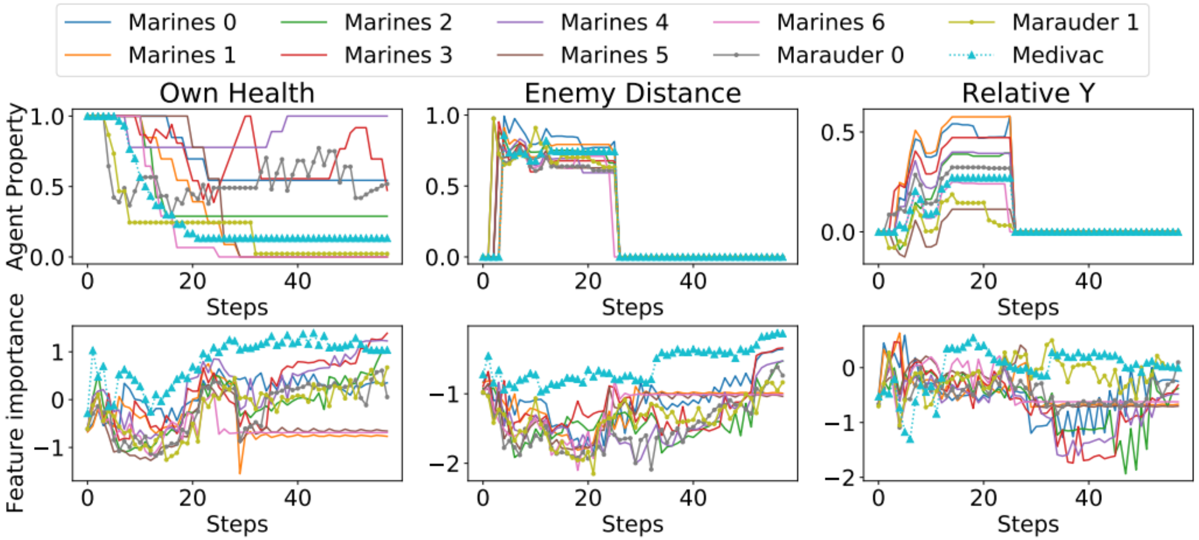
(f) MMM2 (step=35)



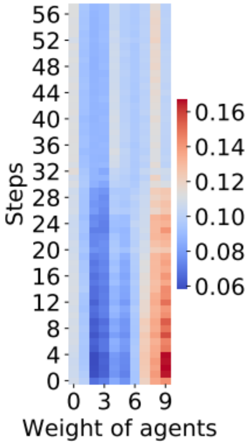
(c) Comparison of feature importance for MIXRTs on 2s3z.



(d) Agent weight heatmap on 2s3z.



(e) Comparison of feature importance for MIXRTs on MMM2.



(f) Agent weight heatmap on MMM2

NA²Q: Neural Attention Additive Model for Interpretable Multi-Agent Q-Learning

Zichuan Liu, Yuanyang Zhu, Chunlin Chen
{zichuanliu, yuanyang}@smail.nju.edu.cn, clchen@nju.edu.cn

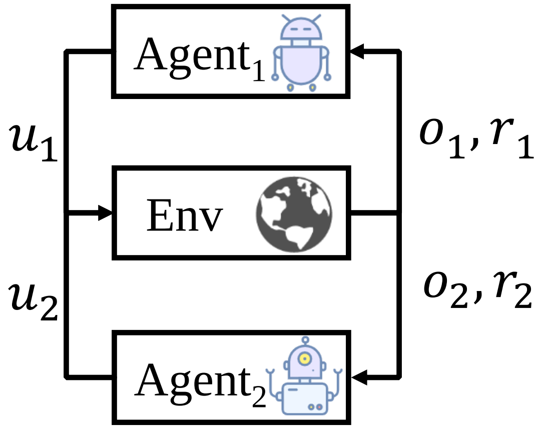
Methodology

- The joint action-value function expands it in terms of Q_i by the Taylor expansion:

$$Q_{tot} = f_0 + \sum_{i=1}^{n} \alpha_i Q_i + \cdots + \sum_{i_1,\ldots,i_l}^{n} \alpha_{i_1 \ldots i_l} \prod_{j=1}^{l} Q_{i_j} + \cdots$$

These issues make intrinsic explanations popular.

Generalized Additive Models



Methodology

- The joint action-value function expands it in terms of Q_i by the Taylor expansion:

$$Q_{tot} = f_0 + \sum_{i=1}^{n} \alpha_i Q_i + \cdots + \sum_{i_1,\ldots,i_l}^{n} \alpha_{i_1 \ldots i_l} \prod_{j=1}^{l} Q_{i_j} + \cdots$$

- We enrich it with an extended GAM method:

$$Q_{tot} = f_0 + \overbrace{\sum_{i=1}^{n} \alpha_i \underbrace{f_i\left(Q_i\right)}_{\text{order-1}}}^{\text{① similar to VDN}} + \cdots + \sum_{k \in \mathcal{D}_l} \alpha_k \underbrace{f_k\left(Q_k\right)}_{\text{order-}l} + \cdots + \overbrace{\alpha_{1\ldots n} \underbrace{f_{1\ldots n}(Q_1, \ldots, Q_n)}_{\text{order-}n}}^{\text{② e.g., QMIX}}$$

Methodology

- The joint action-value function expands it in terms of Q_i by the Taylor expansion:

$$Q_{tot} = f_0 + \sum_{i=1}^{n} \alpha_i Q_i + \cdots + \sum_{i_1,\ldots,i_l}^{n} \alpha_{i_1 \ldots i_l} \prod_{j=1}^{l} Q_{i_j} + \cdots$$

- We enrich it with an extended GAM method:

$$Q_{tot} = f_0 + \overbrace{\sum_{i=1}^{n} \alpha_i \underbrace{f_i\left(Q_i\right)}_{\text{order-1}}}^{\text{1 similar to VDN}} + \cdots + \sum_{k \in \mathcal{D}_l} \alpha_k \underbrace{f_k\left(Q_k\right)}_{\text{order-}l} + \cdots + \overbrace{\alpha_{1\ldots n} \underbrace{f_{1\ldots n}(Q_1, \ldots, Q_n)}_{\text{order-}n}}^{\text{2 e.g., QMIX}}$$

- We provide an approximation guarantee that there always exists an upper bound on our generalization according to regret analysis under the 1-Lipschitz loss approximation:

Theorem B.2. *Let ℓ be 1-Lipschitz, $\delta \in (0,1]$ and Assumption B.1 hold with constants $\{C_1, C_2, \eta\}$. Then, for L_1-norm models, where $\|\boldsymbol{a}_{ld}\|_1 \leq B_a, 1 \leq l \leq n$, and $\|\boldsymbol{\lambda}\|_1 \leq B_\lambda$ where $\boldsymbol{\lambda} = \{\{\lambda_{ld}\}_{d=1}^{\rho_l}\}_{l=1}^{n}$, there exists some absolute constants $\{C_1, C_2\}$ with probability at least $1-\delta, \delta \in (0,1]$ that we have*

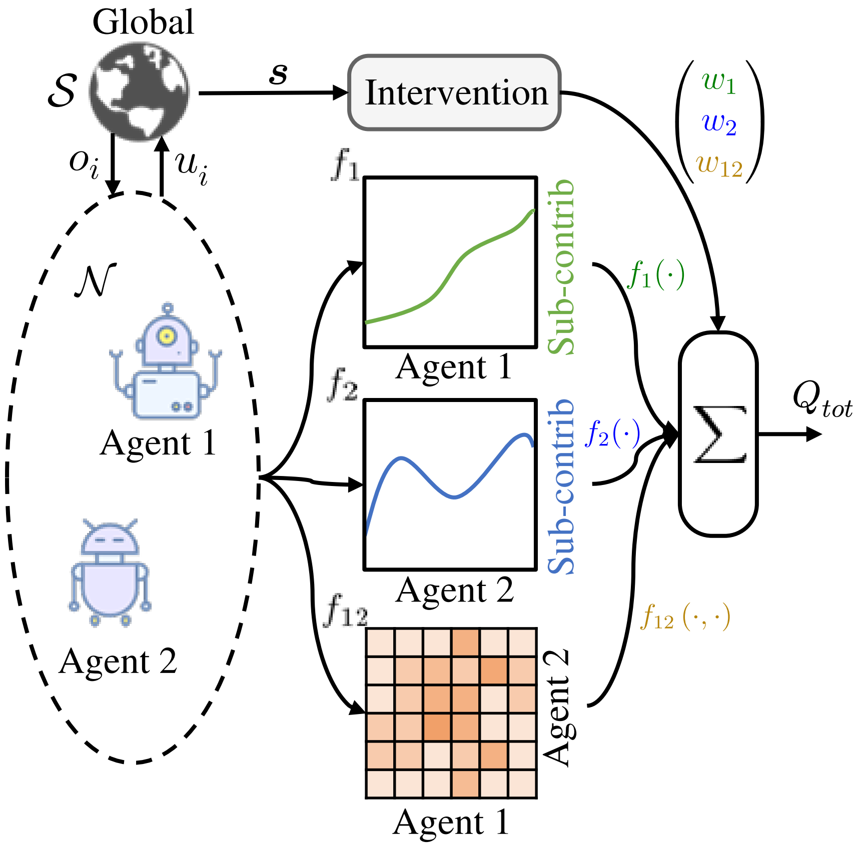
$$\mathcal{L}(\widehat{Q}_{tot}) - \mathcal{L}(Q^\star_{tot}) \leq 2B_\lambda \cdot \left(\sum_{l=1}^{n}(B_a)^l\right)\sqrt{\frac{\log(n)}{b}} + \frac{C_1}{C_2} \cdot \left(\sum_{l=1}^{n} \exp(-\rho_l^\eta)\right) + 2(\sqrt{2}+1) \cdot \sqrt{\frac{\log(2/\delta)}{b}}. \qquad (14)$$

Methodology

- We maintain both the performance and interpretability:

$$Q_{tot} = f_0 + \sum_{i=1}^{n} \alpha_i f_i(Q_i) + \sum_{ij \in \mathcal{D}_2} \alpha_{ij} f_{ij}(Q_i, Q_j)$$

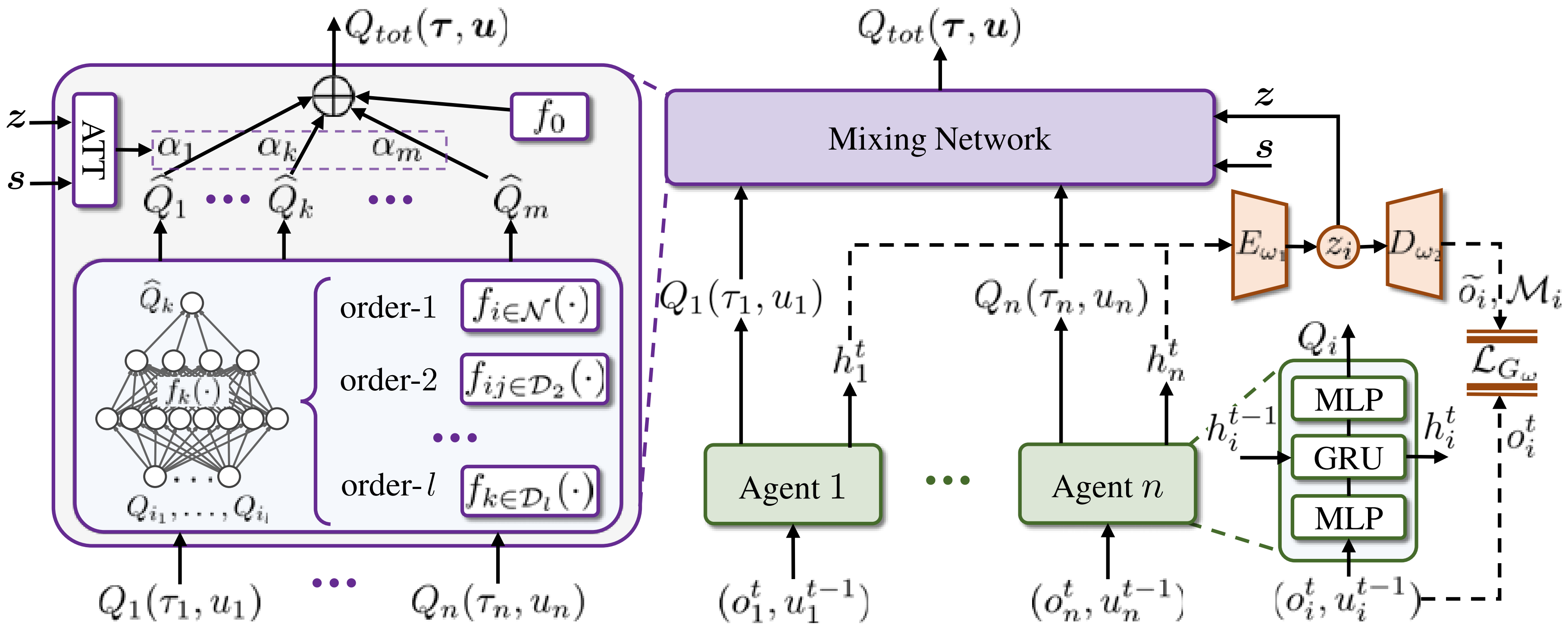
- An example of value decomposition via the GAMs family in MARL. The shape function $f_k \in \{f_1, \cdots, f_{1\ldots n}\}$ learns individual or pairwise action values as a team contribution.



Overall Architecture

A novel subfamily of GAMs named **Neural Attention Additive Q-learning** (NA^{2}Q):

- NA^{2}Q models all possible higher-order interactions of Q-values.
- NA^{2}Q provides local semantics by maximizing the observation resemblance.



Implementation

- Construct the identity semantic to provide a captured interpretation mask:

$$z_i = E_{\omega_1}(\tau_i), \mathcal{M}_i = \varsigma(D_{\omega_2}(z_i)) \xrightarrow{\text{Trained by}} \mathcal{L}_{G_\omega} = \mathcal{L}_{vae} + \sum_{i=1}^{n} \|\mathcal{M}_i\|_1$$

- The credit is computed with the identity semantics and the global state by an attention mechanism as an intervention function:

$$\alpha_k = [\alpha_i, \alpha_{ij}] = \frac{\exp((\boldsymbol{w}_z \boldsymbol{z})^\top \mathrm{ReLU}(\boldsymbol{w}_s \boldsymbol{s}))}{\sum_{k=1}^{m} \exp((\boldsymbol{w}_z \boldsymbol{z})^\top \mathrm{ReLU}(\boldsymbol{w}_s \boldsymbol{s}))}$$

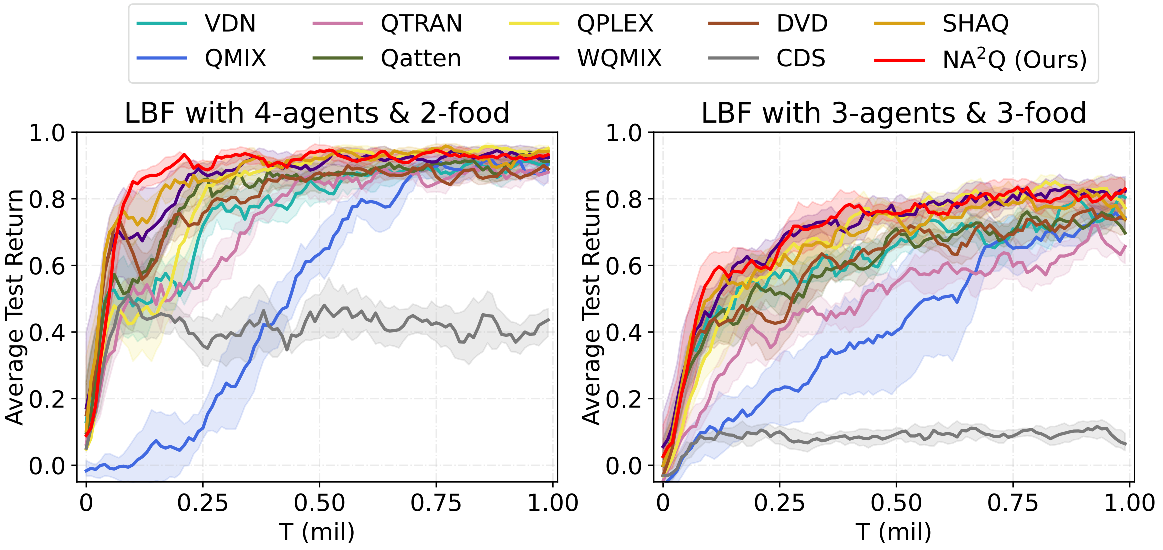
- Transform the local Q-values into a neural GAM paradigm within order-2:

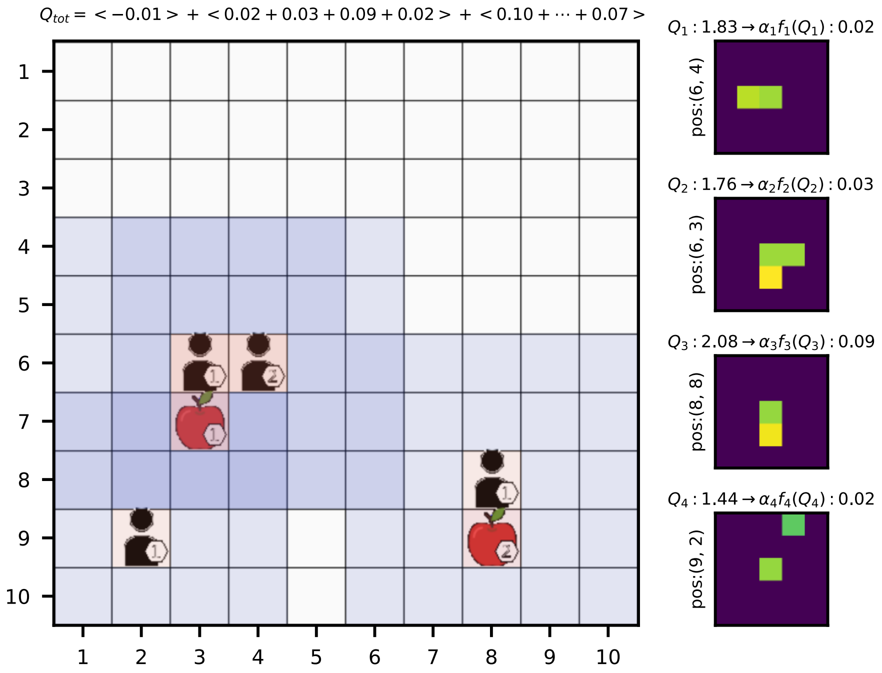
$$Q_{tot} = f_0(\boldsymbol{s}) + \sum_{i=1}^{n} \alpha_i f_i(Q_i) + \sum_{i=1}^{n} \sum_{j>i}^{n} \alpha_{ij} f_{ij}(Q_i, Q_j)$$

Experiments

Level Based Foraging (LBF)

- ➢ Our method achieves competitive performance in LBF tasks, and each agent captures task-relevant semantic information.

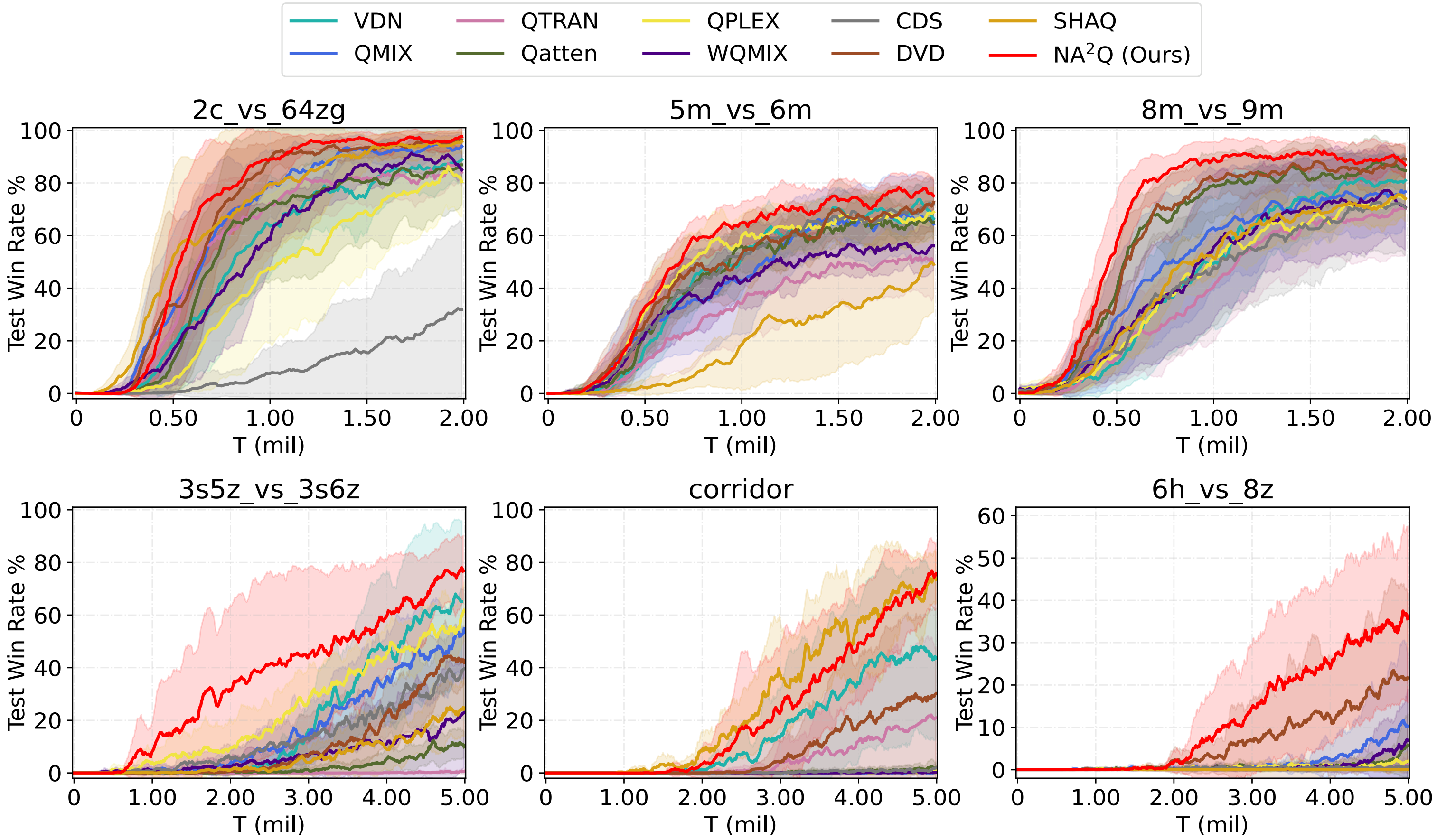




Experiments

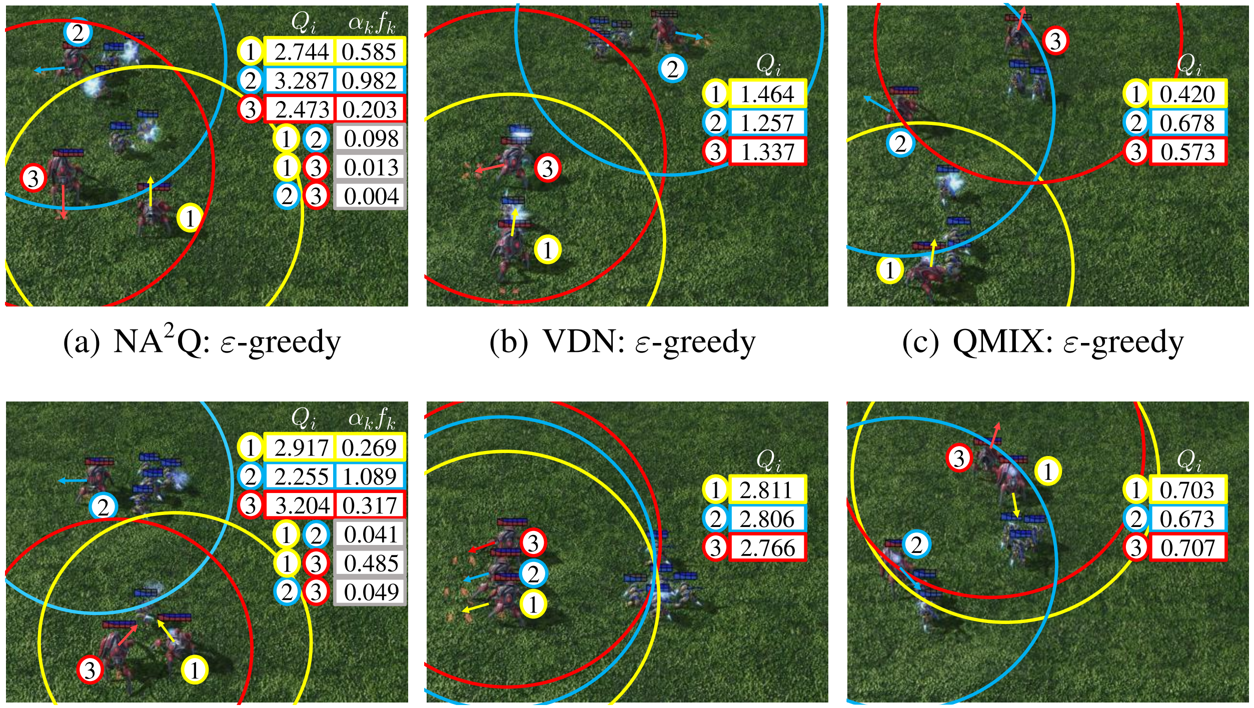
StarCraft Multi-Agent Challenge (SMAC)

➢ NA^{2}Q consistently gains almost the best performance on all scenarios.



Experiments

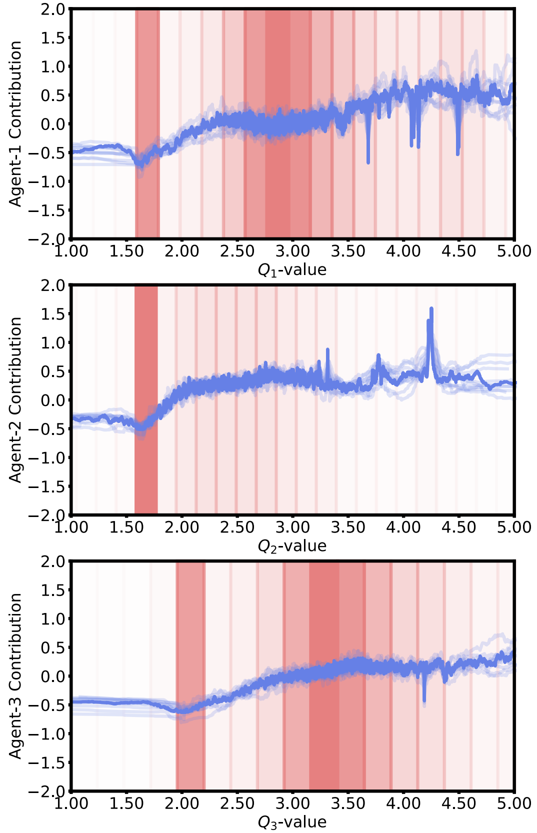
➢ The values of VDN and QMIX are difficult to explain, while the Q-values of the decomposition by NA²Q intend to correspond more clearly to the actions.



(a) NA²Q: ε-greedy (b) VDN: ε-greedy (c) QMIX: ε-greedy

(d) NA²Q: greedy (e) VDN: greedy (f) QMIX: greedy

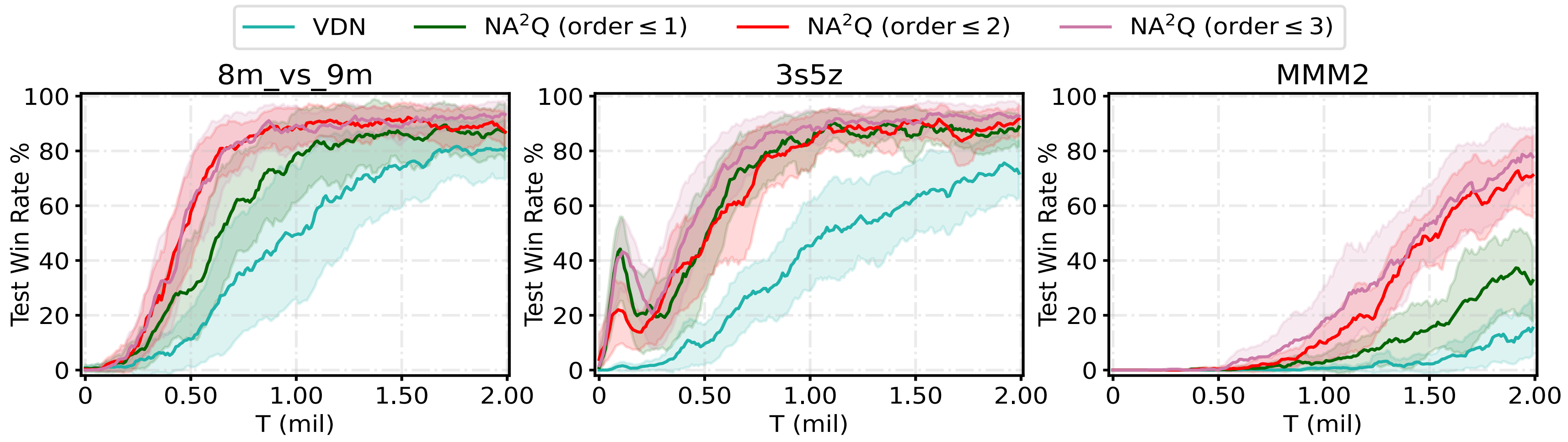
(a) Visualization of evaluation for NA²Q and baselines on 3s vs 5z map. Each decomposed Q-value is displayed at the top-right.



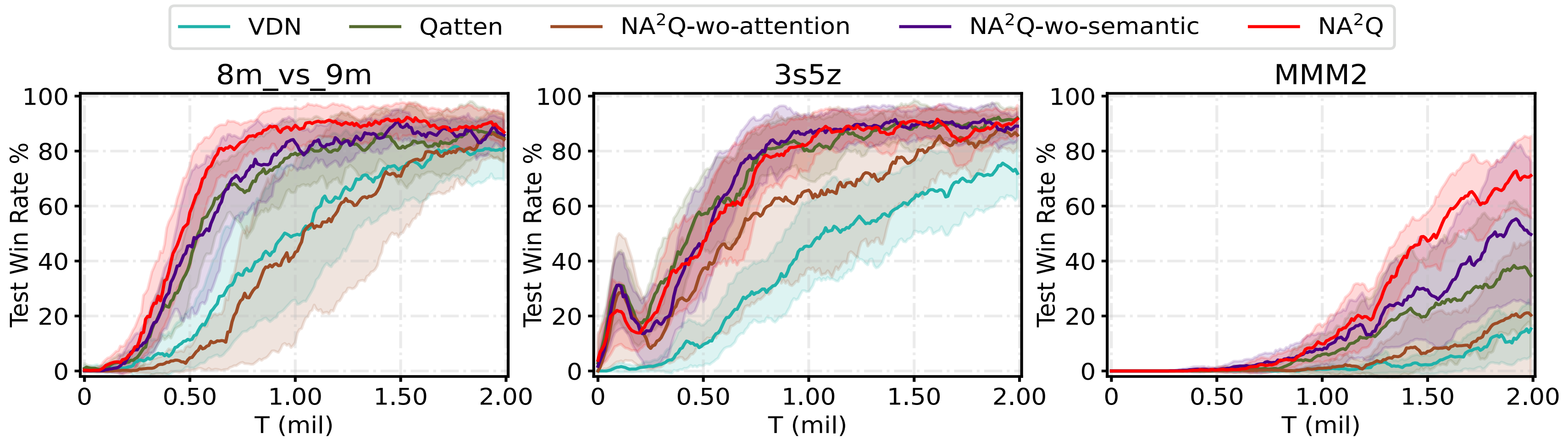
(b) Learned shape functions.

Ablation Study

➢ How does the model's performance benefit from the number of interaction orders and the intervention function?



(a) Number of interactive order terms



(b) Influence of identity semantics and attention

Higher Replay Ratio Empowers Sample-Efficient Multi-Agent Reinforcement Learning

**Linjie Xu, Zichuan Liu, Alexander Dockhorn,
Diego Perez-Liebana, Jinyu Wang, Lei Song, Jiang Bian**

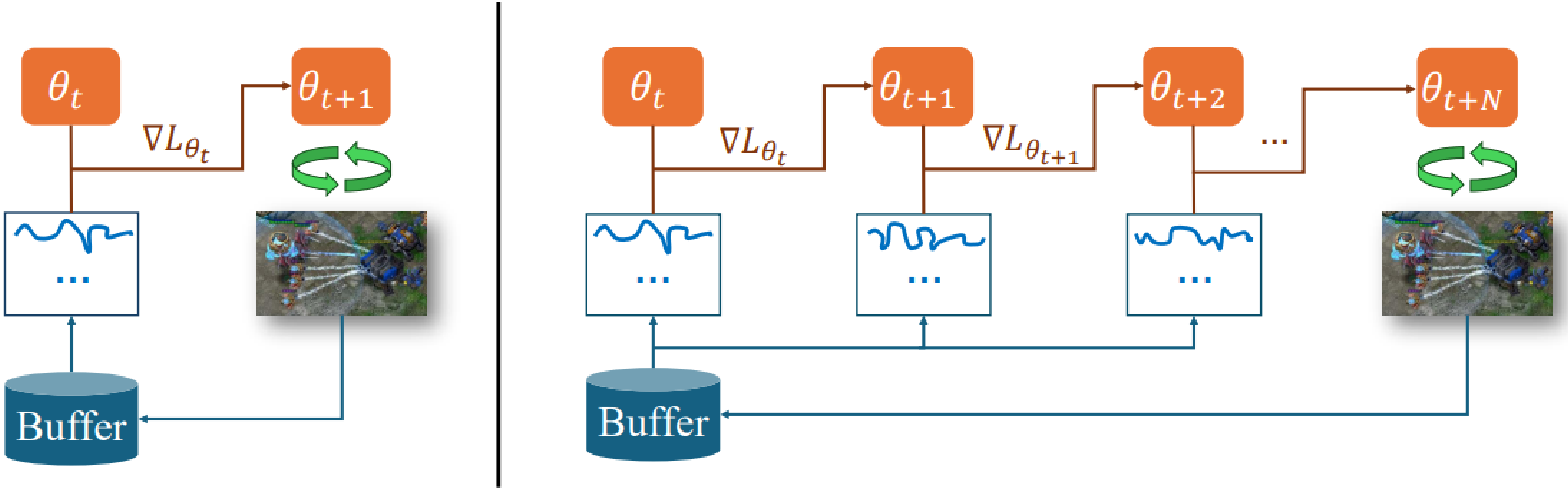
Knowing What Not to Do: Leverage Language Model Insights for Action Space Pruning in Multi-agent Reinforcement Learning

**Zhihao Liu, Xianliang Yang, Zichuan Liu, Yifan Xia, Wei Jiang,
Yuanyu Zhang, Lijuan Li, Guoliang Fan, Lei Song, Bian Jiang**

Higher Replay Ratio

The agent parameter is updated by applying the gradient descent operator:

$$\theta_{t+1} = \theta_t - \alpha_\theta \nabla \mathcal{L}_\theta,$$
$$\phi_{t+1} = \phi_t - \alpha_\phi \nabla \mathcal{L}_\phi,$$

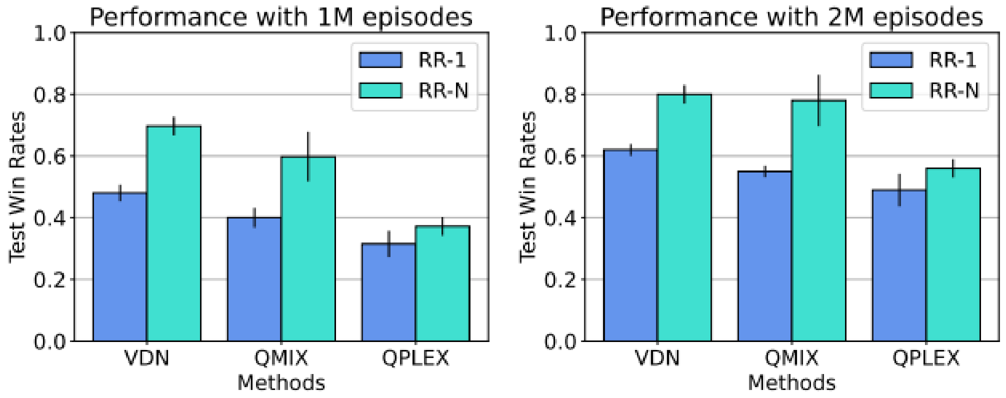


MARL Training

MARL Training with higher replay ratio

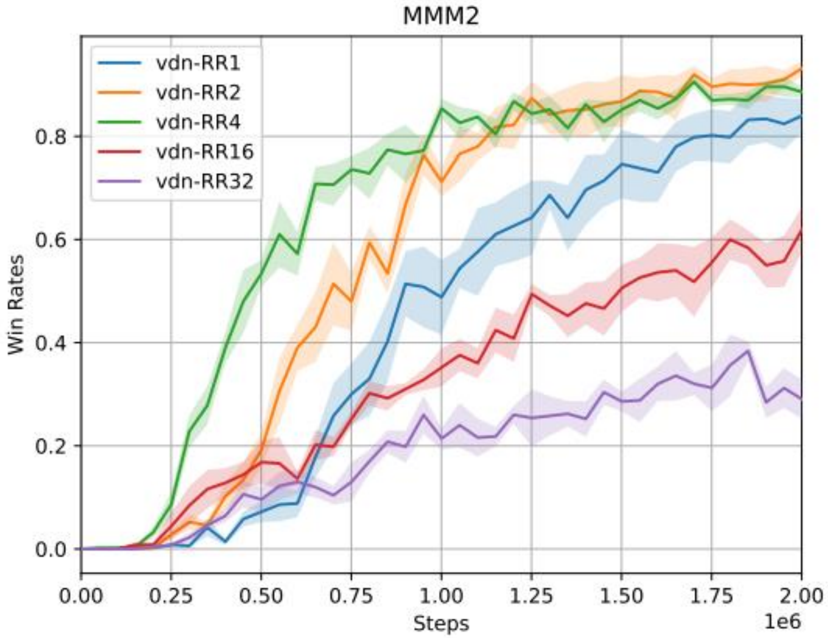
Higher Replay Ratio

Increase the frequency of the gradient updates per environment interaction!



Algorithm 1 MARL Training

Hyperparameters: replay ratio N, learning rate α_θ and α_ϕ, η_θ, η_α
Initialize: Environment $\mathcal{E}$, θ, θ', ϕ , ϕ', $\mathcal{D} = \emptyset$
for $j = 1, 2, \cdots, J$ **do**
  $\tau_j \sim \mathcal{E}$
  $\mathcal{D} = \mathcal{D} \cup \{\tau_j\}$
  for $k = 1, 2, \cdots, N$ **do**
    $\mathcal{B}_k \sim \mathcal{D}$
    $\theta \leftarrow \theta - \alpha_\theta \nabla \mathcal{L}_\theta(\theta, \mathcal{B}_k)$ (Equation 3)
    $\phi \leftarrow \phi - \alpha_\phi \nabla \mathcal{L}_\phi(\phi, \mathcal{B}_k)$ (Equation 4)
    $\theta' \leftarrow (1 - \eta_\theta)\theta' + \eta_\theta \theta$
    $\phi' \leftarrow (1 - \eta_\alpha)\phi' + \eta_\alpha \phi$



LLMs for Action Pruning

Exploration functions:

$$E_1, \ldots, E_K \sim \mathtt{LLM}_c\left(\mathtt{prom}, \mathtt{LLM}_g(\mathtt{prom})\right).$$

Evolutionary search:

$$\phi_1^i, \phi_2^i, \ldots, \phi_K^i \leftarrow \phi_{\text{best}}^{i-1}.$$

Reflection and feedback:

$$\mathtt{prom} \leftarrow \mathtt{prom} : \mathtt{Reflection}(E_{\text{best}}, F_{\text{best}}).$$

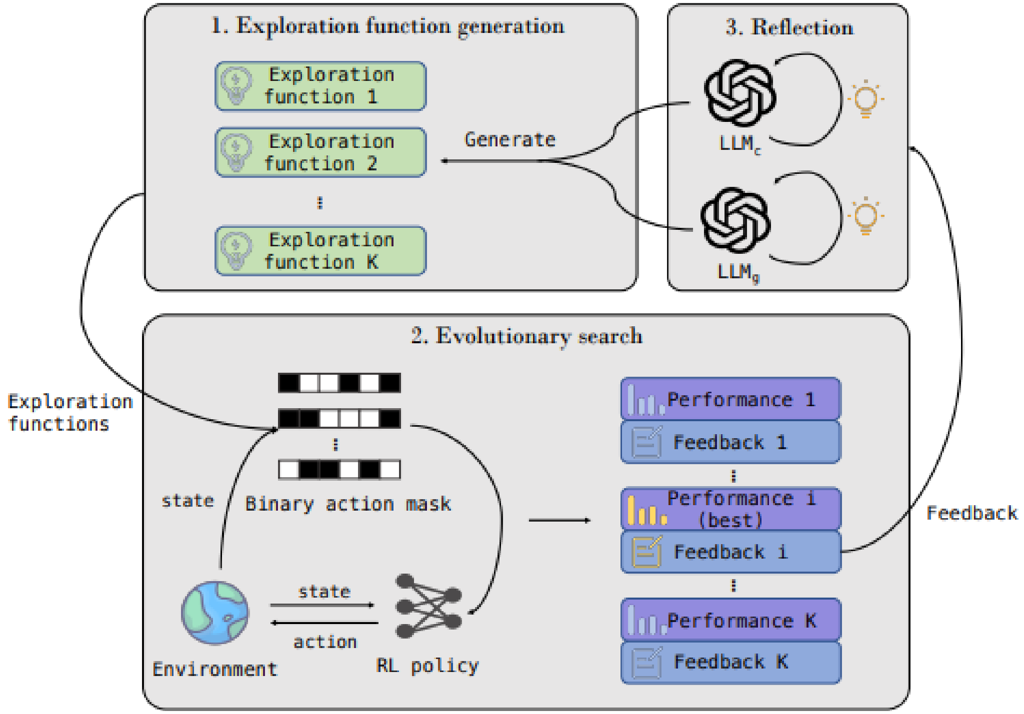


Figure 1: eSpark firstly generates K exploration functions via zero-shot creation. Each exploration function is then used to guide an independent policy, and the evolutionary search is performed to find the best-performing policy. Finally, eSpark reflects on the feedback from the best performance policy, refines, and regenerates the exploration functions for the next iteration.

LLMs for Action Pruning

Table 1: Performance in MABIM, a higher profit indicates a better performance. The "Standard" scenario features a single echelon with sufficient capacity. The "2/3 echelons" involves challenges of multi-echelon cooperation. The "Lower/Lowest" scenarios are the challenges where SKUs compete for insufficient capacity, while "500 SKUs scenarios" assess scalability. The '-' indicates CTDE algorithms are not researched in the scalability challenges.

Method	**Avg. profits (K)** 100 SKUs scenarios					500 SKUs scenarios				
	Standard	2 echelons	3 echelons	Lower	Lowest	Standard	2 echelons	3 echelons	Lower	Lowest
IPPO	690.6	1412.5	1502.9	431.1	287.6	3021.2	7052.0	7945.7	3535.9	2347.4
QTRAN	529.6	1595.3	2012.2	70.1	19.5	-	-	-	-	-
QPLEX	358.9	1580.7	704.2	379.8	259.3	-	-	-	-	-
MAPPO	719.8	1513.8	1905.4	478.3	265.8	-	-	-	-	-
BS static	563.7	1666.6	2338.9	390.7	-1757.5	3818.5	8151.2	11926.3	3115.1	-9063.8
BS dynamic	684.2	1554.2	2378.2	**660.6**	-97.1	4015.7	8399.3	11611.1	**3957.5**	2008.6
(S, s)	737.8	1660.8	1725.2	556.9	203.7	4439.4	**9952.1**	10935.7	3769.3	2212.4
eSpark	**823.7**	**1811.4**	**2598.7**	579.5	**405.0**	**4468.6**	9437.3	**12134.2**	3775.7	**2519.5**

Table 2: Performance in SUMO, including the mean and standard deviation (in parentheses). A lower time indicates a better performance.

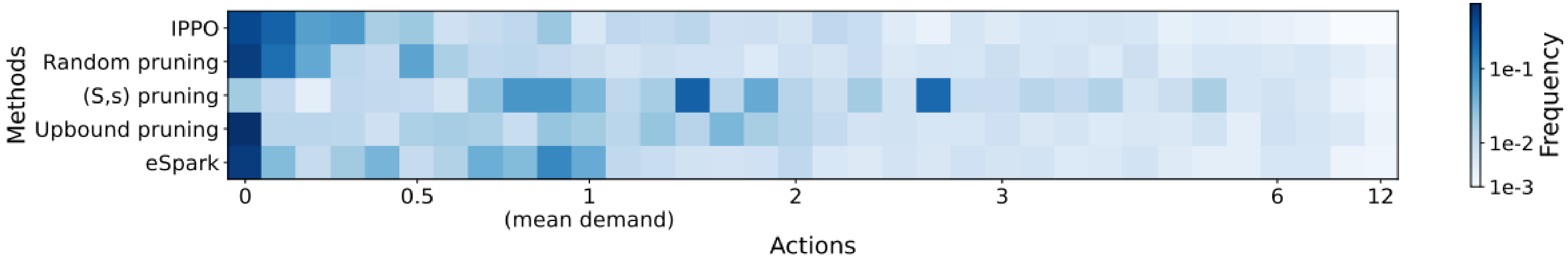
Method	**Avg. delay (seconds)** Grid 4×4	Arterial 4×4	Grid 5×5	Cologne8	Ingolstadt21	**Avg. trip time (seconds)** Grid 4×4	Arterial 4×4	Grid 5×5	Cologne8	Ingolstadt21
FTC	161.14 (3.77)	1229.68 (16.79)	820.88 (24.36)	85.27 (1.21)	224.96 (11.91)	291.48 (4.45)	676.77 (13.70)	584.54 (24.17)	145.93 (0.84)	352.06 (9.29)
MaxPressure	63.39 (1.34)	995.23 (77.02)	242.85 (16.04)	31.63 (0.61)	228.64 (15.83)	174.68 (2.05)	702.09 (23.61)	269.35 (9.62)	95.67 (0.62)	352.30 (15.06)
IPPO	48.40 (0.45)	884.73 (38.94)	228.78 (11.59)	27.60 (1.70)	342.97 (43.61)	160.12 (0.60)	506.18 (10.39)	238.03 (7.10)	91.41 (1.60)	464.50 (43.30)
MAPPO	51.25 (0.58)	958.43 (181.72)	958.43 (181.72)	32.55 (4.66)	347.59 (47.59)	160.01 (0.54)	757.40 (242.00)	247.56 (3.71)	94.31 (1.77)	480.66 (49.46)
CoLight	53.40 (1.89)	**820.67 (58.65)**	339.66 (48.55)	27.48 (1.30)	296.47 (106.82)	165.77 (1.89)	**470.33 (12.34)**	305.41 (44.43)	91.66 (1.29)	410.59 (97.29)
MPLight	63.51 (0.64)	1142.98 (79.65)	223.44 (16.18)	37.93 (0.45)	**196.74 (9.88)**	172.47 (0.89)	583.21 (45.84)	255.49 (6.26)	110.56 (1.18)	**331.42 (11.79)**
eSpark	**48.36 (0.32)**	854.22 (68.21)	**209.49 (13.98)**	**25.39 (1.27)**	243.92 (15.81)	**159.74 (0.44)**	484.87 (58.21)	**235.20 (6.80)**	**89.50 (1.36)**	367.57 (15.03)

LLMs for Action Pruning

Comparison of exploration functions before and after editing:

```python
def compute_mask(agent_states: AgentStates, supply_chain: SupplyChain, action_space: list) -> np.ndarray:
    warehouse_list = supply_chain.get_warehouse_list()
    num_sku = len(agent_states)
    total_mask = np.ones((len(warehouse_list), num_sku, len(action_space)))

    for i, warehouse_name in enumerate(warehouse_list):
        capacity = supply_chain[warehouse_name, "capacity"]
        # Get the volume for all sku
        sku_volume = agent_states[warehouse_name, "volume", "today", "all_skus"]
        # Get the in_stock for all sku
        in_stock = agent_states[warehouse_name, "in_stock", "today", "all_skus"]
        # Get the mean demand for all the skus in the past lookback period
        history_demand_mean = np.average(agent_states[warehouse_name, "demand", "lookback", "all_skus"], axis=1)

        for a, action in enumerate(action_space):
            # Calculate the potential replenishment for all SKUs
            replenishment = history_demand_mean * action
            # If the replenishment will make the total volume of all SKUs exceeds the capacity of the warehouse, mask this action
            if np.sum(sku_volume * (in_stock + replenishment)) > capacity:
                total_mask[i, :, a] = 0

            # Add an additional condition to mask actions that bring in too much replenishment
            if action > 2:
            if action > 1.5:
                total_mask[i, :, a] = 0

            # Add another condition to encourage more exploration in the early training stages
            if agent_states.current_step < agent_states.durations * 0.3 and action < 0.5:
                total_mask[i, :, a] = 0

    return total_mask, {}
```

Action selection frequency for IPPO and various pruning methods on the 100 SKUs Lowest scenario.



Closing Remarks

- We use a recursive soft decision tree to model the decision-making process of a single agent, and apply it in MARL
- We present NA^2Q which combines the inherent interpretability of GAMs into Multi-Agent Reinforcement Learning.
- We explore sample efficiency in multi-agent buffer pools using replay radio in the MARL system
- We use large language models for action space pruning and reconstruction of agents in MARL

Thanks for your listening!

Any Questions? Please use the chat !