Dynamic Sight Range Selection in Multi-Agent Reinforcement Learning

Wei-Chen Liao
Department of Computer Science, National Yang Ming Chiao Tung University
Hsinchu, Taiwan
wcl.cs11@nycu.edu.tw

Ti-Rong Wu
Institute of Information Science, Academia Sinica
Taipei, Taiwan
tirongwu@iis.sinica.edu.tw

I-Chen Wu
Department of Computer Science, National Yang Ming Chiao Tung University
Hsinchu, Taiwan
icwu@cs.nycu.edu.tw(correspondence)

ABSTRACT

Multi-agent reinforcement Learning (MARL) is often challenged by the sight range dilemma, where agents either receive insufficient or excessive information from their environment. In this paper, we propose a novel method, called *Dynamic Sight Range Selection (DSR)*, to address this issue. DSR utilizes an Upper Confidence Bound (UCB) algorithm and dynamically adjusts the sight range during training. Experiment results show several advantages of using DSR. First, we demonstrate using DSR achieves better performance in three common MARL environments, including Level-Based Foraging (LBF), Multi-Robot Warehouse (RWARE), and StarCraft Multi-Agent Challenge (SMAC). Second, our results show that DSR consistently improves performance across multiple MARL algorithms, including QMIX and MAPPO. Third, DSR offers suitable sight ranges for different training steps, thereby accelerating the training process. Finally, DSR provides additional interpretability by indicating the optimal sight range used during training. Unlike existing methods that rely on global information or communication mechanisms, our approach operates solely based on the individual sight ranges of agents. This approach offers a practical and efficient solution to the sight range dilemma, making it broadly applicable to real-world complex environments.

KEYWORDS

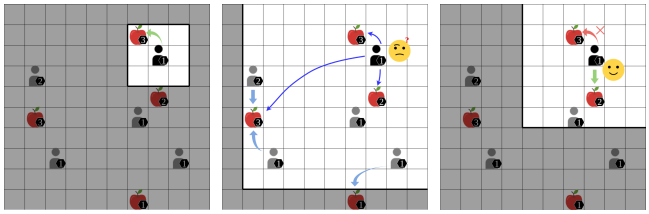
Multi-Agent Reinforcement Learning, Sight Range Dilemma, Upper Confidence Bound (UCB)

ACM Reference Format:
Wei-Chen Liao, Ti-Rong Wu, and I-Chen Wu. 2025. Dynamic Sight Range Selection in Multi-Agent Reinforcement Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning [26] has achieved significant success in various domains, such as gaming [20], circuit design [12], and recommendation systems [1]. To extend its applicability to complex real-world problems, particularly those involving multiple agents that must cooperate or compete to achieve shared goals, multi-agent reinforcement learning (MARL) [13] has recently emerged to address the challenges in multi-agent environments, including multiplayer gaming [28], autonomous vehicles [35], robotic control [31], and traffic signal control [14, 36].

Various approaches have been developed in cooperative MARL. For instance, independent learning (IL) techniques, such as Independent Q-Learning (IQL) and Independent Proximal Policy Optimization (IPPO) [5, 21], train each agent individually, treating other agents as part of the environment. While this simplifies the learning process by reducing the complexity of considering all agents simultaneously, it also introduces the challenge of non-stationarity, where the learning dynamics continuously change as other agents adapt during training [33]. On the other hand, Centralized Training with Decentralized Execution (CTDE) has been a widely used framework for cooperative MARL problems, addressing non-stationarity issues by allowing agents to access all available information during centralized training while maintaining decentralized decision-making. Successful methods based on CTDE include Q-learning-based approaches like QMIX [18], QTRAN [23], and QPLEX [29], as well as policy optimization methods like MAPPO [34] and HAPPO [10].



(a) (b) (c)

Figure 1: An illustration of the sight range dilemma in the level-based foraging game, where players are required to cooperate to collect food. (a) With a small sight range, the player may fail to see other players. (b) With a large sight range, the player receives excessive information irrelevant to their decision. (c) With an appropriate sight range, the player can easily identify the right partner to cooperate with in collecting food.

Due to the partial observability in MARL, one of the key challenges is limited information that each agent can observe from the environment, often referred to as the *sight range*. In real-world applications, agents usually perceive only a limited portion of the environment due to constraints such as sensor range. For instance, in autonomous driving, each vehicle can detect only a small region

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

of its surroundings rather than the entire environment. A small sight range generally provides insufficient information, making it difficult for agents to make effective decisions. However, previous research suggests that a larger sight range is not always better, as it often includes excessive and irrelevant information, which hinders the learning process and leads to worse performance [8, 22]. The trade-off in selecting an appropriate sight range, known as the *sight range dilemma* [22], remains a critical challenge in MARL, as illustrated in Figure 1. Previous works [8, 22] have addressed this dilemma, mainly focusing on communication mechanisms that use self-attention to leverage global information, enabling each agent to identify relevant agents and adjust its sight range accordingly. Although these methods mitigate the sight range dilemma, they rely on acquiring global information for all agents, which is often impractical in real-world environments.

To tackle this challenge, we propose **Dynamic Sight Range Selection (DSR)**, a novel approach that directly addresses the sight range dilemma without requiring global information. Specifically, DSR dynamically adjusts the sight range during training using an Upper Confidence Bound (UCB) algorithm [7], allowing the agent to be trained with varying sight ranges and converge on the most suitable sight range. Unlike previous methods that rely on global information, DSR controls only the sight ranges of individual agents, offering clearer insights into how much information is needed in different environments. Moreover, DSR can automatically discover the optimal sight range, which is particularly valuable for real-world applications. For instance, in autonomous systems, the sight range obtained through DSR can guide sensor design, helping to balance sight range, reduce costs, and maintain high performance.

The contributions of this paper are summarized as follows:

- DSR effectively addresses the sight range dilemma issue and outperforms the baseline model without DSR in three common MARL environments, including LBF, RWARE, and SMAC.
- By training agents with different sight ranges, DSR also accelerates the training process.
- DSR can be simply integrated with any MARL algorithm such as QMIX and MAPPO.
- The dynamically selected sight ranges further provide additional interpretability and explainability.

2 BACKGROUND

2.1 Multi-agent Reinforcement Learning

Cooperative multi-agent problems can be modeled as Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) [4, 15], defined as $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \mathcal{Z}, \mathcal{R}, \gamma \rangle$, where $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ is the set of agents, $\mathcal{S}$ is the set of global states, $\mathcal{A}$ is the set of joint individual actions from each agent, $\mathcal{P}$ is the state transition function, $\mathcal{O}$ is the set of observations, $\mathcal{Z}$ is the observation function, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor for calculating future rewards. Given a state s^t at timestep t, where s^t represents the global state with full observability, each agent $n_i \in \mathcal{N}$ receives a partial observation $o_i^t \in \mathcal{O}$, which is derived from the observation function $\mathcal{Z}(s^t, n_i)$. The agent then selects an action $a_i \in \mathcal{A}$ based on its policy $\pi_i(a_i \mid \tau_i)$, where $\tau_i = \{o_i^1, a_i^1, \ldots, o_i^t\}$ is the action-observation history for agent n_i. The joint action $a = \{a_1, a_2, \ldots, a_N\}$ is formed by the actions selected by all agents, and the environment transitions to the next state s^{t+1} according to the state transition function $\mathcal{P}(s^{t+1} \mid s^t, a)$. All agents receive a shared reward r^t from the reward function $\mathcal{R}(s^t, a)$ and obtain a new partial observation o_i^{t+1}. This process repeats continuously until the termination.

The goal of Dec-POMDPs is to find the optimal joint policy of all agents, π^*, to maximize the global value function, $Q^\pi(s^t, a^t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s^t, a^t) \mid s_0 = s, \pi\right]$. Due to the partial observability in Dec-POMDPs, $Q(\tau, a)$ is often used instead of $Q(s, a)$, where τ represents the history of observations and actions of all agents. Namely, agents are supposed to make decisions based on incomplete information and learn an optimal policy π_i that maximizes the total expected reward while cooperating with other agents in a partially observable environment.

2.2 Centralized Training with Decentralized Execution (CTDE)

The *Centralized Training with Decentralized Execution (CTDE)* framework has been widely adopted to address the challenges of partial observability in MARL [18, 23, 25, 29, 30, 34]. CTDE utilizes centralized training, where agents have access to global states during the learning process. However, during execution, decision-making is decentralized, and each agent relies only on its observation. This approach allows agents to make independent decisions during execution while benefiting from coordinated training to enhance overall team performance.

In CTDE, value-based methods often employ the *Individual-Global-Max (IGM)* principle, which ensures consistency between the optimal joint action and the individual actions of each agent. This principle is expressed as:

$$\arg\max_a Q(s, a) = \left(\arg\max_{a_1} Q_1(\tau_1, a_1), \ldots, \arg\max_{a_N} Q_N(\tau_N, a_N)\right),$$

where $Q(s, a)$ is the joint Q-function for all agents, and $Q_i(\tau_i, a_i)$ is the local Q-function for agent n_i. This ensures that during execution, each agent can independently select actions that align with the global objective. On the other hand, actor-critic methods in CTDE directly optimize agents' policies. These methods use centralized critics during training, which have access to the global state s or information about other agents, while each agent maintains its own decentralized actor for execution [6, 11, 34].

2.3 Sight Range Dilemma in MARL

Recent studies have investigated the impact of sight range on agent coordination within multi-agent reinforcement learning, particularly in relation to communication mechanisms. For instance, MASIA [8] highlights the issue of redundant global information in multi-agent systems, which can hinder effective coordination among agents. Their experiments in the Traffic Junction [24] environment illustrate that agents utilizing the QMIX algorithm perform better with a limited sight range compared to super-limited or full-sight settings. CAMA [22], which focuses on dynamic team composition problems, introduces the sight range dilemma and employs attention weights to selectively choose entities to focus on, combined with messages from a global coach. It demonstrates

that both excessive and insufficient information can negatively impact learning. However, previous works mainly focus on communication environments and utilize communication mechanisms to acquire global information. In contrast, this paper aims to investigate whether it is possible to address the sight range dilemma directly without the need for additional communication systems.

2.4 Upper Confidence Bound

The *Upper Confidence Bound (UCB)* [3] algorithm is a widely used method to balance exploration and exploitation in decision-making problems. At each timestep t, UCB selects the action with the highest estimated reward by considering both the empirical mean reward $\hat{X}_t$ and the upper confidence bound of the reward U_t, calculated as follows:

$$UCB_t(a_i) = \hat{X}_t(a_i) + c \times U_t(a_i) = \hat{X}_t(a_i) + c \times \sqrt{\frac{\log t}{N_t(a_i)}}, \quad (1)$$

where $\hat{X}_t(a_i)$ is the empirical mean reward for action a_i up to time t, $N_t(a_i)$ is the number of times a_i has been selected, and c is a hyperparameters that controls the exploration and exploitation. The first term encourages exploitation by favoring actions with higher average rewards, while the second term promotes exploration by choosing actions that have been less frequently explored.

To extend UCB to non-stationary decision-making environments where the optimal choice may change over time, non-stationary UCB algorithms [7] have been proposed. Specifically, the *sliding-window upper confidence bound (SW-UCB)* algorithm modifies the calculation of the empirical mean reward in the original UCB by focusing on a limited window of recent observations rather than the entire history:

$$\begin{aligned} UCB_t^{SW}(a_i) &= \hat{X}_t^{SW}(a_i) + c \times U_t^{SW}(a_i), \\ &= \frac{1}{N_t(a_i, w)} \sum_{j=t-w+1}^{t} r_j(a_i) \mathbb{1}_{\{UCB_j^{SW}=i\}} \\ &\quad + c \times \sqrt{\frac{\log \min(t, w)}{N_t(a_i, w)}}, \text{ and} \\ N_t(a_i, w) &= \sum_{j=t-w+1}^{t} \mathbb{1}_{\{UCB_j^{SW}=i\}}, \end{aligned} \quad (2)$$

where $N_t(a_i, w)$ is the number of times a_i has been selected within the sliding window, $r_j(a_i)$ represents the reward received from selecting action a_i at timestep j, and w is a hyperparameter for the size of sliding window. By limiting the mean reward to the most recent observations, SW-UCB balances adapting to the latest results while maintaining exploration. This approach makes SW-UCB well-suited for dynamic, non-stationary environments.

3 DYNAMIC SIGHT RANGE SELECTION

In this section, we present our approach, named *Dynamic Sight Range Selection (DSR)*, which dynamically adjusts and finds the most suitable sight range for all agents. The overview of the DSR method is illustrated in Figure 2, which can be incorporated into any MARL algorithm. In our design, DSR is built upon the Dec-POMDP but modifies the *observation function*, which processes the observation within a given sight range, and introduces a *meta-controller*, which determines the sight range. Each component is described in detail below.

3.1 Observation Function

The observation function $\mathcal{Z}(s, n_i)$, as reviewed in subsection 2.1, is modified to $\mathcal{Z}(s, n_i, d)$ to incorporate a given sight range d, which limits the portion of global state s that each agent n_i can observe. The sight range d can be defined based on the requirements of each specific environment. For instance, in Figure 1, the sight range is defined as the distance in grid cells that an agent can observe around its current position, with $d = 1$, $d = 6$, and $d = 3$ for Figure 1a, 1b, and 1c, respectively. In real-world applications, the sight range may not always be a fixed distance around the agent, and a larger d does not necessarily cover smaller d. For example, in autonomous driving cars, different values of d can represent different sensor designs with varying coverage shapes. Under this setting, our goal is to find the suitable sight range d during training.

3.2 Meta-Controller

Given a set of sight ranges $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ with a total of M possible ranges, a meta-controller Λ is incorporated into the MARL training by selecting a sight range $d \in \mathcal{D}$ for the agents at the beginning of each episode. The learning process can be viewed as hierarchical optimization, where the meta-controller dynamically adjusts and selects the optimal sight range at the start of each episode, while the MARL algorithm focuses on maximizing the global value function Q^π within the selected sight range throughout the episode. As both the meta-controller and MARL agents learn simultaneously, we adapt the SW-UCB algorithm for the meta-controller to balance exploration and exploitation during training. The calculation of the UCB score at episode e for each sight range d_i follows equation (2) as:

$$\begin{aligned} UCB_e^{\Lambda}(d_i) &= \hat{X}_e(d_i) + c \times U_e(d_i) \\ &= \frac{1}{N_e(d_i, w)} \sum_{j=e-w+1}^{e} r_j(d_i) \mathbb{1}_{\{UCB_j^{\Lambda}=i\}} \\ &\quad + c \times \sqrt{\frac{\log \min(e, w)}{N_e(d_i, w)}}, \end{aligned} \quad (3)$$

where e represents the e-th episode, w is the sliding window size, $r_j(d_i)$ is the episode return with sight range d_i in the j-th episode, and $N_e(d_i, w)$ is the number of games that have been played with sight range d_i within the sliding window. After training, the meta-controller converges to an optimal sight range. During execution, we simply choose the sight range d with the maximum average return in the window, $\arg\max(\bar{r}(d_i))$, where $\bar{r}(d_i)$ is the average return for each sight range d_i.

3.3 Training Algorithm

Specifically, we summarize the training process in the Algorithm 1. In line 4, the meta-controller selects the current best sight range d_e^* based on the UCB at the start of each episode. Once the sight range d_e^* is selected, any MARL algorithm can be applied to train agents for one episode. The observation is modified based on the selected sight range and used by the agents during interactions with the

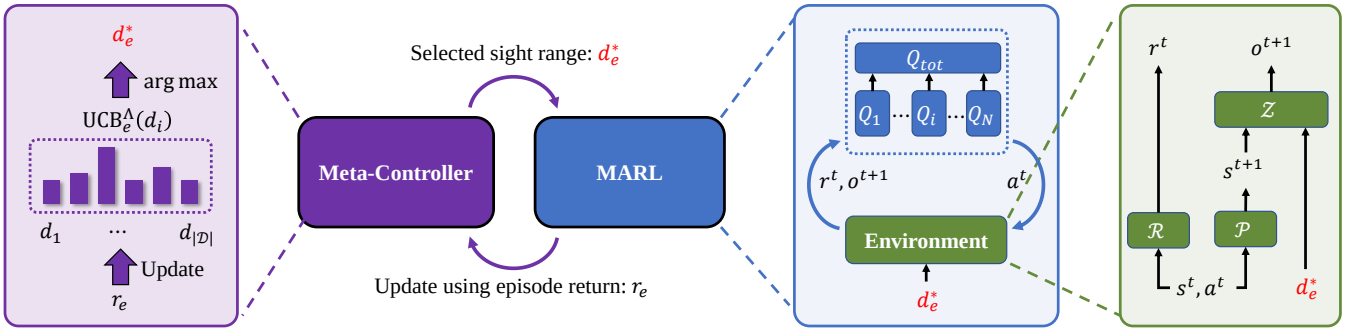


Figure 2: Overview of the Dynamic Sight Range Selection framework. The meta-controller (left) dynamically selects the current optimal sight range d_e^* using the sliding-window UCB based on the episode return r_e. The selected sight range d_e^* is used in the MARL training (right), where agents interact with the environment, receiving observations within the selected sight range.

environment, as shown in line 5 to 7. The modified observation is stored in the buffer associated with the MARL algorithm. This approach allows the agent to learn from different sight ranges simultaneously, facilitating exploration in the early stages. As the meta-controller converges, the replay buffer will gradually accumulate more samples with optimized sight ranges.

After training for one episode, the episode return r_e is obtained, and the statistics of $N_e(d_e^*, w)$ and reward for the meta-controller are updated, as shown in line 8 to 10. These updated statistics are then used for the next episode. Overall, the algorithm operates as a hierarchical optimization process, where both the meta-controller and the MARL algorithm are learning simultaneously. The meta-controller optimizes the sight range selection, while the MARL algorithm focuses on maximizing agent performance within the chosen sight range, leading to efficient coordination and better long-term performance. Moreover, this design allows the meta-controller to easily integrate with any MARL algorithm.

Algorithm 1 Dynamic Sight Range Selection (DSR)

1: **Input:** Set of sight ranges $\mathcal{D}$, sliding window size w, constant c, total number of training episodes E, total number of training steps per episode T
2: **Output:** The best sight range d^*
3: **for** episode $e = 1$ to E **do**
4: Meta-controller selects sight range d_e^*:

$$d_e^* = \arg\max_{d_i \in \mathcal{D}} \hat{X}_e(d_i) + c \times U_e(d_i)$$

5: Generate one episode with a modified observation function $\mathcal{Z}(s, n_i, d_e^*)$
6: Obtain the episode return r_e
7: Train the episode by any MARL algorithm
8: **if** $d_{e-t} \neq d_e^*$ **then**

$$N_e(d_e^*, w) = N_e(d_e^*, w) + 1$$

9: **end if**
10: Record reward r_e for d_e^*
11: **end for**

4 EXPERIMENT

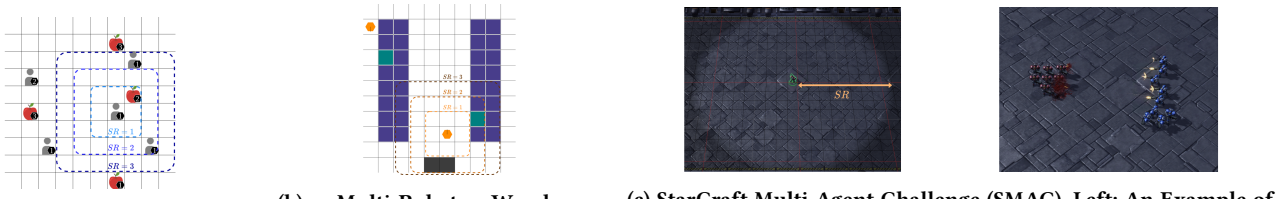
4.1 Experiment Setup

We build DSR upon the training framework EPyMARL [16] and conduct experiments in three common MARL environments, as illustrated in Figure 3, including Level-Based Foraging (LBF) [2, 16], Multi-Robot Warehouse (RWARE) [16], and the StarCraft Multi-Agent Challenge (SMAC) [19]. For the SW-UCB in the meta-controller, we choose $c = 2$ for the exploration coefficient and $w = 5000$ for the sliding window size. The following paragraphs describe each environment and its specific settings.

Level-Based Foraging (LBF). LBF is a grid-based multi-agent environment designed to evaluate cooperative behavior, where agents must collaborate to collect food in a grid world, as shown in Figure 3a, making it a common benchmark for testing MARL algorithms. The episode return ranges from 0 to 1 and is proportional to the fraction of food collected relative to the total food score on the map. The sight range is defined by the distance (in grid cells) that an agent can observe around its current position. For example, a sight range of 1 provides a 3x3 ($1+1\times2=3$) view around the agent, while a sight range of 2 expands this to a 5x5 ($1+2\times2=5$) area. In LBF, the notation 10x10-4p-2f-coop represents a 10x10 grid map, four players (4p), two food items (2f), and a cooperative mode (coop), where agents must cooperate more intensively to collect the food. For our experiments, we choose three settings: 10x10-4p-2f-coop, 10x10-4p-2f, and 10x10-4p-4f-coop. For each setting, we select two different default sight ranges, $d = 6$ and $d = 10$. Note that in a 10x10 grid map, a sight range of $d = 10$ allows each agent to observe the entire map, equivalent to having access to the global state. For the DSR method, the meta-controller can select the sight range from the set $\mathcal{D}$ as follows:

- $\mathcal{D} = \{2, 4, 6\}$ for $d = 6$
- $\mathcal{D} = \{2, 4, 6, 8, 10\}$ for $d = 10$

Multi-Robot Warehouse (RWARE). RWARE is a grid-based multi-agent environment where cooperative robots work together to transport goods within a warehouse. Agents must locate and deliver requested shelves (girds marked in teal) to the designated locations (grids marked in black at the bottom), and then return the shelves to empty positions before continuing with the next



(a) Level-Based Foraging (LBF)

(b) Multi-Robot Warehouse (RWARE)

(c) StarCraft Multi-Agent Challenge (SMAC). Left: An Example of sight range; right: a game example (8m_vs_9m)

Figure 3: MARL environments used in our experiments.

delivery, as shown in Figure 3b. Each successful delivery yields a reward of +1, incentivizing agents to complete delivery cycles. Since both RWARE and LBF are grid-based environments, the sight range in RWARE is defined in the same way as LBF. In RWARE, we use two map sizes, a 10x11 grid map (tiny) and a 10x20 (small) map, each with two agents (denoted as tiny-2ag and small-2ag). For each setting, we select two different default sight ranges, $d = 3$ and $d = 5$. For the DSR method, the meta-controller can select the sight range from the set $\mathcal{D}$ as follows:

- $\mathcal{D} = \{1, 2, 3\}$ for $d = 3$
- $\mathcal{D} = \{1, 2, 3, 4, 5\}$ for $d = 5$

StarCraft Multi-Agent Challenge (SMAC). SMAC is a well-established benchmark for multi-agent reinforcement learning, focusing on micromanagement tasks in StarCraft II. Each agent controls a unique unit, and the goal is to defeat AI-controlled opponents in combat, as shown in Figure 3c. The sight range is defined as the visibility radius around each unit. We select six settings for SMAC, including 5m_vs_6m, 8m_vs_9m, 10m_vs_11m, 3s_vs_5z, 3s5z, and MMM2. For each setting, we select three different default sight ranges, $d = 9$, $d = 15$, and $d = 21$. For the DSR method, the meta-controller can select the sight range from the set $\mathcal{D}$ as follows:

- $\mathcal{D} = \{3, 6, 9\}$ for $d = 9$
- $\mathcal{D} = \{3, 6, 9, 12, 15\}$ for $d = 15$
- $\mathcal{D} = \{3, 6, 9, 12, 15, 18, 21\}$ for $d = 21$

In LBF and RWARE, the global state s provided by the environment does not contain all information. This mirrors real-world applications, where obtaining complete global information is often impractical [9, 27]. A common approach is to approximate the global state by concatenating the observations from all agents. Although the original SMAC environment provides complete global information, our focus is on how sight ranges influence the learning complexity of observations and states. Therefore, we adopt a variant of SMAC where the global state is derived by concatenating the observations from all agents, the same approach used in LBF and RWARE.

4.2 Performance of DSR

We first train QMIX [18], a common MARL algorithm, with and without DSR across the three environments with various settings described in the previous subsection. Each setting was trained with five different seeds.

Table 1 shows the results comparing DSR and baseline (without DSR) across different environment settings. For LBF and RWARE, we use the mean test return, while for SMAC, we use the mean test win rate. Note that the last notation after the hyphen (e.g., the "10s" in 10x10-4p-2f-coop-10s) represents the default sight range used in the baseline (without DSR). For DSR, the set of sight ranges is as mentioned in the previous subsection.

Our results show that DSR consistently improves performance across all three environments. In LBF, our method significantly outperforms the baseline, with substantial improvement in complex cooperative settings such as 10x10-4p-4f-coop-6s (0.772 compared to 0.277) and 10x10-4p-4f-coop-10s (0.798 compared to 0.338). This result demonstrates the sight range dilemma, where larger sight ranges in complex environments are not always beneficial, as agents may receive excessive irrelevant information that negatively impacts decision-making, as illustrated in Figure 1. The scores for LBF range from 0 to 1, so in some simple settings both w/ and w/o DSR can achieve nearly optimal scores. However, even in these cases, DSR significantly accelerates the training process, as shown in Figure 4. For more challenging LBF tasks, such as 10x10-4p-4f-coop-10s, which require deeper cooperation, DSR demonstrates substantial improvements (0.798 vs. 0.338) compared to training without DSR. For RWARE, both models struggle in the small map, likely due to the map's difficulty, achieving relatively lower scores, but DSR outperforms the baseline on the tiny map. In SMAC, DSR consistently enhances performance, particularly in scenarios with larger sight range settings, such as 3s_vs_5z-21s (0.712 compared to 0.292) and MMM2-21s (0.714 compared to 0.19). In conclusion, these results demonstrate the effectiveness of dynamically selecting sight ranges using DSR, highlighting its robustness across a wide range of environments.

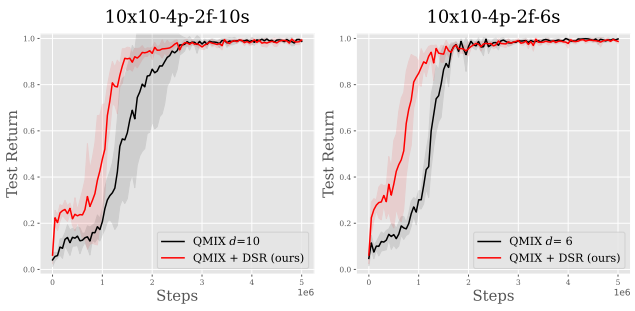


Figure 4: Mean test returns on two LBF environment settings. The shaded area represents the standard deviation.

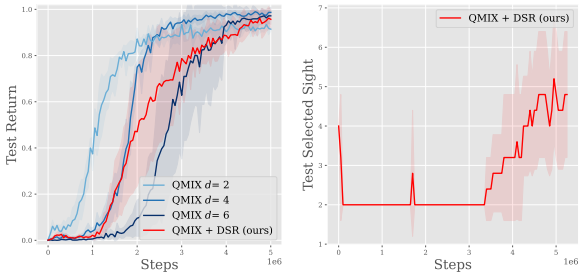
Table 1: Comparison between baseline and DSR across three environments.

		w/o DSR	w/ DSR (ours)
LBF	10x10-4p-2f-coop-10s	0.769 ± 0.384	**0.925 ± 0.064**
	10x10-4p-2f-coop-6s	**0.972 ± 0.037**	0.957 ± 0.040
	10x10-4p-2f-10s	0.988 ± 0.005	**0.993 ± 0.004**
	10x10-4p-2f-6s	**0.998 ± 0.004**	0.987 ± 0.010
	10x10-4p-4f-coop-10s	0.338 ± 0.339	**0.798 ± 0.050**
	10x10-4p-4f-coop-6s	0.277 ± 0.194	**0.772 ± 0.068**
RWARE	tiny-2ag-5s	1.486 ± 1.361	**4.762 ± 4.702**
	tiny-2ag-3s	5.846 ± 1.426	**11.900 ± 6.474**
	small-2ag-5s	**0.074 ± 0.148**	0.050 ± 0.100
	small-2ag-3s	**0.182 ± 0.359**	0.036 ± 0.072
SMAC	5m_vs_6m-9s	0.102 ± 0.028	**0.128 ± 0.066**
	5m_vs_6m-15s	0.080 ± 0.100	**0.140 ± 0.055**
	5m_vs_6m-21s	0.054 ± 0.079	**0.112 ± 0.053**
	8m_vs_9m-9s	0.452 ± 0.114	**0.508 ± 0.077**
	8m_vs_9m-15s	0.234 ± 0.287	**0.504 ± 0.087**
	8m_vs_9m-21s	0.120 ± 0.190	**0.472 ± 0.078**
	10m_vs_11m-9s	0.402 ± 0.220	**0.540 ± 0.061**
	10m_vs_11m-15s	0.122 ± 0.202	**0.546 ± 0.127**
	10m_vs_11m-21s	0.186 ± 0.262	**0.680 ± 0.066**
	3s_vs_5z-9s	**0.716 ± 0.143**	0.676 ± 0.065
	3s_vs_5z-15s	0.498 ± 0.409	**0.660 ± 0.081**
	3s_vs_5z-21s	0.292 ± 0.260	**0.712 ± 0.110**
	3s5z-9s	**0.854 ± 0.086**	**0.854 ± 0.051**
	3s5z-15s	**0.808 ± 0.094**	0.770 ± 0.075
	3s5z-21s	**0.784 ± 0.156**	0.736 ± 0.069
	MMM2-9s	0.690 ± 0.122	**0.736 ± 0.076**
	MMM2-15s	0.572 ± 0.113	**0.648 ± 0.094**
	MMM2-21s	0.190 ± 0.266	**0.714 ± 0.101**

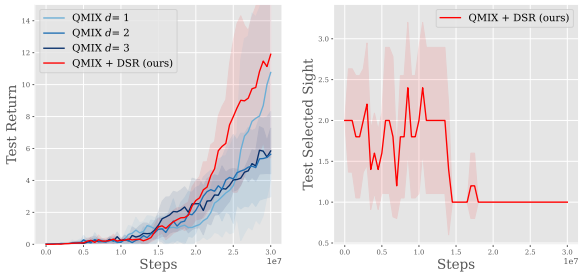
In addition, we found that using DSR can accelerate the training process. For example, Figure 4 shows the training curves for two LBF settings, including 10x10-4p-2f-10s and 10x10-4p-2f-6s. While both DSR and the baseline converge to similar results by the end of the training, the training curve of DSR (red) rises more quickly during the early training steps and remains higher or equal to the baseline (black) throughout the training process. This demonstrates an additional benefit of using DSR, as training with different sight ranges helps agents learn the game more quickly.

4.3 The Sight Range Dilemma

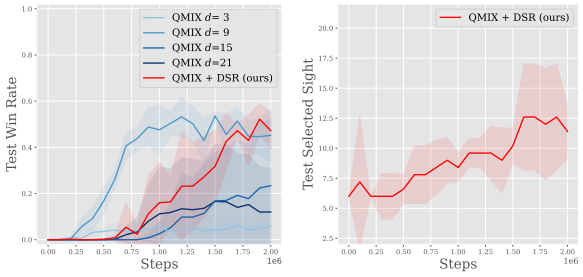
To further investigate how sight range affects training, we conduct an additional experiment where the baseline uses different fixed sight ranges throughout the entire training process. We select one setting for each environment, as illustrated in Figure 5. Figure 5a shows the results for the LBF environment. The left subfigure shows the mean test return curves of the baseline with different fixed sight ranges ($d = 2, 4,$ and 6) and our DSR approach, while the right subfigure shows the sight range dynamically selected by DSR during the training process. From the left subfigure, we observe that using a smaller sight range results in higher returns at the early training. For instance, at training step 2×10^6, the performance of $d = 2$ is better than $d = 4$, and $d = 4$ outperforms $d = 6$. However, by the end of the training, $d = 2$ plateaus, while $d = 4$ and $d = 6$ achieve better performance. Interestingly, since our DSR can dynamically adjust the sight range, we observe that DSR tends to select smaller sight ranges at the beginning and gradually shifts to larger sight ranges, as shown in the right subfigure. The RWARE environment, as shown in Figure 5b, shows another interesting result. In the left subfigure, we observe that using $d = 1$ performs well in this environment setting. Surprisingly, in the right subfigure, DSR initially attempts to explore a larger sight range but then quickly converges to $d = 1$. In the SMAC environment, as shown in figure 5c, we observe that DSR gradually selects sight range from $d = 6$ to $d = 12$, and the results outperform all fixed sight ranges. This suggests that agents benefit from training with smaller sight ranges initially, which helps them transition more effectively to larger sight ranges during training.



(a) LBF 10x10-4p-2f-coop-6s



(b) RWARE tiny-2ag-3s



(c) SMAC 8m_vs_9m-21s

Figure 5: Experiments on three environment settings. For each subfigure, the left shows the mean test returns, while the right side displays the selected sights by DSR during training.

Moreover, in LBF, we observe that DSR usually starts by selecting smaller sight ranges and then gradually selects larger ones. A

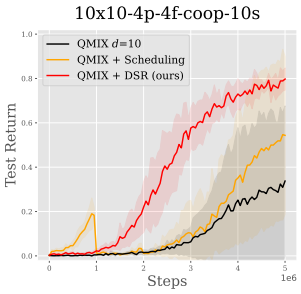


Figure 6: Comparison between the baseline, DSR, and a fixed sight range scheduling approach in the LBF environment.

straightforward approach would be to design a fixed sight range scheduling, such as the sight range progressively expanding from small to large (e.g., $d = 2, 4, 6, 8,$ and 10) at regular intervals, dividing the training steps into five equal phases. As shown in Figure 6, the fixed sight range scheduling approach also performs better than the baseline but still does not outperform DSR. Additionally, when switching sight ranges in the fixed schedule, we observe a noticeable drop in performance. In contrast, DSR allows for smoother sight range adjustments and results in a more stable training curve.

In summary, our results demonstrate that DSR can not only accelerate the training but also automatically discover the appropriate sight range without the need to manually train across all sight ranges. In many real-world complex environments, different sight ranges may yield varying outcomes and it is often difficult to know the best sight range for every task. Therefore, DSR offers an efficient solution for both finding the optimal sight range and subtly addressing the sight range dilemma.

4.4 DSR in Other MARL Algorithms

Since DSR does not modify the underlying MARL algorithms, we conduct experiments to verify its generalizability across different MARL algorithms, including IQL, VDN, IPPO, and MAPPO, in LBF and RWARE environments. In general, the sight range dilemma persists across all algorithms, and DSR consistently outperforms the baseline, similar to the results shown for QMIX (subsection 4.2).

Figure 7 shows one of the results for the LBF environment with the 10x10-4p-4f-coop setting, where DSR significantly outperforms the baseline across all four algorithms. In addition, we observe that DSR follows a similar pattern of adjusting sight ranges across most algorithms, except for VDN, typically starting with smaller sight ranges and gradually transitioning to larger ones. These experiments further demonstrate the robustness and versatility of DSR, showing that our approach can be easily integrated into any MARL algorithm and enhance learning performance without requiring algorithm-specific modifications.

4.5 Exploring Hyperparameters in DSR

In this subsection, we analyze the effectiveness of different hyperparameters in DSR within the LBF environment, including the exploration constant c, the sliding window size w, and various combinations of sight ranges available for the meta-controller to select.

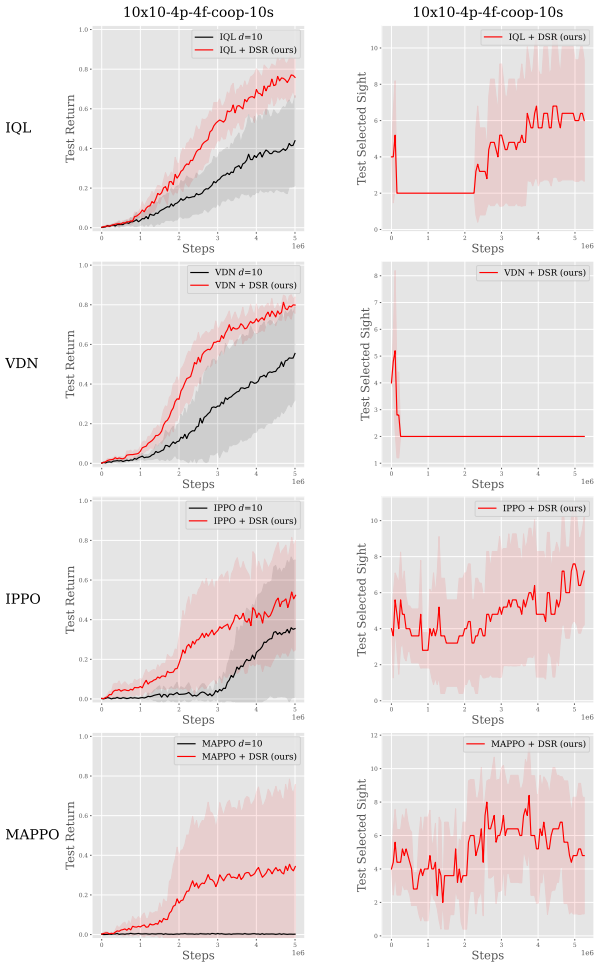
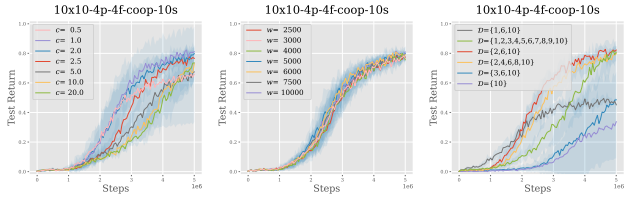


Figure 7: Experiments with four additional MARL algorithms. Top: Mean test returns for the baseline and DSR; Bottom: Sight ranges selected by DSR during training.

The results are shown in Figure 8. For the exploration coefficient c, we observe that neither smaller nor larger values consistently lead to better results, as shown in Figure 8a. The findings suggest that the optimal range for c lies between 1 and 2.5. In contrast, the choice of sliding window size does not significantly affect the results, as shown in 8b.

Next, we analyze the performance when different sets of sight ranges are used in DSR, as shown in Figure 8c. When comparing $\mathcal{D} = \{1, 2, 3, \ldots, 10\}$ (green) and $\mathcal{D} = \{2, 4, 6, 8, 10\}$ (yellow), the results show that a larger set of options for meta-controller slows down the training process, as it increases the complexity of sight range selection for meta-controller and requires more time for agents to adapt. For $\mathcal{D} = \{2, 6, 10\}$ (red), $\mathcal{D} = \{1, 6, 10\}$ (gray), and $\mathcal{D} = \{3, 6, 10\}$ (blue), the results show that the red line outperforms the other two. This is because $d = 2$ enables faster learning during the early stage in LBF, as illustrated in Figure 5a. However, in practice, it is difficult to predict which sight range is best in

advance. In summary, a larger set of sight ranges is less likely to miss potentially effective sight ranges, but it may increase training time due to exploration. On the other hand, a smaller sight range can learn faster if it includes an appropriate sight range, but it also risks omitting better ones. Therefore, designing an appropriate set of sight range $\mathcal{D}$ is important.



(a) Exploration coefficient c. **(b) Sliding window size w.** **(c) Different set of sight range $\mathcal{D}$.**

Figure 8: Experiments for different hyperparameters in the LBF environment.

4.6 Comparison to Communication-based Methods

Finally, we compare our DSR approach to CAMA, a communication-based method that uses an attention-weight ranking mechanism to select relevant entities, aiming to address the sight range dilemma. To evaluate DSR, we integrate it into the source code provided in CAMA [22]. We then follow CAMA's training process using the IM-Qatten algorithm, which combines Qatten [32] with an inverse model [17]. In addition, we use the same environment as CAMA, SMAC Dynamic Team Composition (SMAC-DT), a variant of SMAC where both the number and types of units change dynamically across episodes. The number of agents is randomly set between 3 and 5 during both training and testing. The default sight range in CAMA is set to $d = 9$. For the DSR method, the meta-controller can select the sight range from the set $\mathcal{D} = \{3, 6, 9\}$.

Figure 9 shows that DSR outperforms the attention-weight ranking mechanism across three SMAC-DT environment settings. DSR also provides explanability, as it explicitly indicates which sight range is selected, while CAMA's attention-weight ranking can only show the chosen entities without offering insight into the overall observation strategy. Overall, this experiment demonstrates the advantages of DSR in both performance and interpretability.

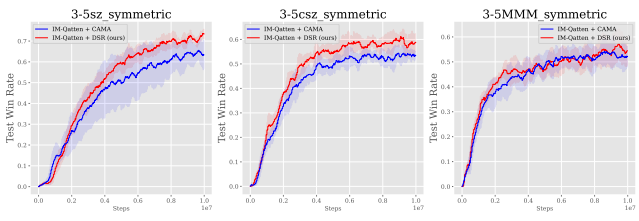


Figure 9: Comparison between CAMA and DSR in the SMAC-DT environment.

5 DISCUSSION

This paper proposes a novel approach named Dynamic Sight Range Selection (DSR), which successfully addresses the sight range dilemma issues. Our experiments show that DSR outperforms the baseline model without using DSR across three MARL environments, including LBF, RWARE, and SMAC. Unlike traditional methods that require manually finding the suitable sight range, DSR automatically identifies the optimal sight range for each environment. In addition, DSR accelerates the training process by gradually shifting the sight range from a smaller to a large one. It further provides interpretability by offering insights into the selected sight ranges. Finally, the method can be seamlessly integrated with multiple MARL algorithms without requiring algorithm-specific modifications.

Future work could explore how DSR generalizes to environments with different domains. For more complex environments, such as those with continuous observation spaces, the design of the meta-controller may require further adjustments. One possible approach is to discretize continuous sight ranges into representative discrete sets for selection.

The insights provided by DSR can also help sensor design in practical applications by balancing sight range, performance, and cost. In addition, in our work, all agents share the same sight range, but future research could investigate using different sight ranges for individual agents, which is particularly relevant for heterogeneous agent settings where agents have different roles or capabilities. In scenarios with a large number of options, the meta-controller may require further modifications to handle this issue effectively. Furthermore, deeper exploration is needed to understand how the structure of an environment influences the optimal sight range, providing new insights into the relationship between observation and environment dynamics.

ACKNOWLEDGEMENT

This research is partially supported by the National Science and Technology Council (NSTC) of the Republic of China (Taiwan) under Grant Number NSTC 113-2221-E-A49-127, NSTC 113-2221-E-A49-128, NSTC 113-2634-F-A49-004, and NSTC 113-2221-E-001-009-MY3. The authors would like to thank Kuo-Hao Ho and Chiu-Chou Lin for some discussions in the early stages.

REFERENCES

[1] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement Learning Based Recommender Systems: A Survey. *ACM Comput. Surv.* 55, 7 (2022), 145:1–145:38. https://doi.org/10.1145/3543846

[2] Stefano V. Albrecht and Subramanian Ramamoorthy. 2013. A Game-Theoretic Model and Best-Response Learning Method for Ad Hoc Coordination in Multiagent Systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '13)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1155–1156.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2 (May 2002), 235–256. https://doi.org/10.1023/A:1013689704352

[4] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. 2000. The Complexity of Decentralized Control of Markov Decision Processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 32–37.

[5] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? https://doi.org/10.48550/arXiv.2011.09533 arXiv:2011.09533 [cs]

[6] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (April 2018). https://doi.org/10.1609/aaai.v32i1.11794

[7] Aurélien Garivier and Eric Moulines. 2008. On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems. https://doi.org/10.48550/arXiv.0805.3415 arXiv:0805.3415 [math, stat]

[8] Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. 2022. Efficient Multi-agent Communication via Self-supervised Information Aggregation. *Advances in Neural Information Processing Systems* 35 (Dec. 2022), 1020–1033. https://proceedings.neurips.cc/paper_files/paper/2022/hash/075b2875e2b671ddd74aeec0ac9f0357-Abstract-Conference.html

[9] Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Xiaodan Liang, Zhihui Li, Xiaojun Chang, and Yaodong Yang. 2023. MARLlib: A Scalable and Efficient Multi-agent Reinforcement Learning Library. *Journal of Machine Learning Research* 24, 315 (2023), 1–23. http://jmlr.org/papers/v24/23-0378.html

[10] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2022. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=EcGGFkNTxdJ

[11] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html

[12] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. 2021. A Graph Placement Methodology for Fast Chip Design. *Nature* 594, 7862 (June 2021), 207–212. https://doi.org/10.1038/s41586-021-03544-w

[13] Zepeng Ning and Lihua Xie. 2024. A Survey on Multi-Agent Reinforcement Learning and Its Application. *Journal of Automation and Intelligence* 3, 2 (June 2024), 73–91. https://doi.org/10.1016/j.jai.2024.02.003

[14] Mohammad Noaeen, Atharva Naik, Liana Goodman, Jared Crebo, Taimoor Abrar, Zahra Shakeri Hossein Abad, Ana L. C. Bazzan, and Behrouz Far. 2022. Reinforcement Learning in Urban Network Traffic Signal Control: A Systematic Literature Review. *Expert Systems with Applications* 199 (Aug. 2022), 116830. https://doi.org/10.1016/j.eswa.2022.116830

[15] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-28929-8

[16] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. https://openreview.net/forum?id=cIrPX-Sn5n

[17] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-Driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2778–2787. https://proceedings.mlr.press/v70/pathak17a.html

[18] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 4295–4304. https://proceedings.mlr.press/v80/rashid18a.html

[19] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2186–2188.

[20] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. 2020. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature* 588, 7839 (Dec. 2020), 604–609. https://doi.org/10.1038/s41586-020-03051-4

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://arxiv.org/abs/1707.06347v2

[22] Jianzhun Shao, Hongchang Zhang, Yun Qu, Chang Liu, Shuncheng He, Yuhang Jiang, and Xiangyang Ji. 2023. Complementary Attention for Multi-Agent Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 30776–30793. https://proceedings.mlr.press/v202/shao23b.html

[23] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 5887–5896. https://proceedings.mlr.press/v97/son19a.html

[24] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/hash/55b1927fdafef39c48e5b73b5d61ea60-Abstract.html

[25] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085–2087.

[26] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[27] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. 2021. Pettingzoo: Gym for Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems* 34 (2021), 15032–15043.

[28] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature* 575, 7782 (Nov. 2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z

[29] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Rcmk0xxIQV

[30] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 9876–9886. https://proceedings.mlr.press/v119/wang20f.html

[31] Yuchen Xiao, Weihao Tan, and Christopher Amato. 2022. Asynchronous Actor-Critic for Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (Dec. 2022), 4385–4400. https://proceedings.neurips.cc/paper_files/paper/2022/hash/1c153788756d35559c22d105d1182c30-Abstract-Conference.html

[32] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. https://doi.org/10.48550/arXiv.2002.03939 arXiv:2002.03939

[33] Yaodong Yang and Jun Wang. 2020. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. https://arxiv.org/abs/2011.00583v3

[34] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems* 35 (Dec. 2022), 24611–24624. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html

[35] Ruiqi Zhang, Jing Hou, Florian Walter, Shangding Gu, Jiayi Guan, Florian Röhrbein, Yali Du, Panpan Cai, Guang Chen, and Alois Knoll. 2024. Multi-Agent Reinforcement Learning for Autonomous Driving: A Survey. https://doi.org/10.48550/arXiv.2408.09675 arXiv:2408.09675 [cs]

[36] Haiyan Zhao, Chengcheng Dong, Jian Cao, and Qingkui Chen. 2024. A Survey on Deep Reinforcement Learning Approaches for Traffic Signal Control. *Engineering Applications of Artificial Intelligence* 133 (July 2024), 108100. https://doi.org/10.1016/j.engappai.2024.108100