GTDE: Grouped Training with Decentralized Execution for Multi-agent Actor-Critic

Mengxian Li[1,2], **Qi Wang**[1,2*], **Yongjun Xu**[1,2]

[1]Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
limengxian24b@ict.ac.cn, wangqi08@ict.ac.cn

Abstract

The rapid advancement of multi-agent reinforcement learning (MARL) has given rise to diverse training paradigms to learn the policies of each agent in the multi-agent system. The paradigms of decentralized training and execution (DTDE) and centralized training with decentralized execution (CTDE) have been proposed and widely applied. However, as the number of agents increases, the inherent limitations of these frameworks significantly degrade the performance metrics, such as win rate, total reward, etc. To reduce the influence of the increasing number of agents on the performance metrics, we propose a novel training paradigm of grouped training decentralized execution (GTDE). This framework eliminates the need for a centralized module and relies solely on local information, effectively meeting the training requirements of large-scale multi-agent systems. Specifically, we first introduce an adaptive grouping module, which divides each agent into different groups based on their observation history. To implement end-to-end training, GTDE uses Gumbel-Sigmoid for efficient point-to-point sampling on the grouping distribution while ensuring gradient backpropagation. To adapt to the uncertainty in the number of members in a group, two methods are used to implement a group information aggregation module that merges member information within the group. Empirical results show that in a cooperative environment with 495 agents, GTDE increased the total reward by an average of 382% compared to the baseline. In a competitive environment with 64 agents, GTDE achieved a 100% win rate against the baseline.

Code — https://github.com/lemonsinx/GTDE

1 Introduction

Multi-agent reinforcement learning has recently made remarkable progress across diverse real-world applications, such as multi-robot cooperative tasks (Krnjaic et al. 2022), complex games (Silver et al. 2018; Vinyals et al. 2019), computer network (Wang et al. 2022; Wang, Huang, and Xu 2024), and autonomous driving (Zhou et al. 2022; Dinneweth et al. 2022). These scenarios are often constrained by partial observability or limited communication, necessitating the use of decentralized policies. These policies rely solely on the observation history of each individual agent to make decisions. The decentralized policies also avoid the issue of exponential growth in the action space as the number of agents increases, making a fully centralized policy impractical for large-scale multi-agent systems. The decentralized policies can be learned through either Decentralized Training and Execution (DTDE) and Centralized Training with Decentralized Execution (CTDE) (Albrecht, Christianos, and Schäfer 2024).

DTDE utilizes a decentralized approach to simplify the training of multi-agents by using a single-agent RL algorithm for each individual agent. However, this training paradigm introduces two inevitable issues. Firstly, agents are unable to leverage information from other agents in the environment, making cooperation difficult (Mguni et al. 2023). Secondly, since each agent treats other agents as part of the environment, they are affected by environmental non-stationary caused by updates in the policies of other agents (Laurent, Matignon, and Fort-Piat 2011), which intensifies with a larger number of agents.

CTDE uses a centralized approach to train decentralized policies (Oliehoek, Spaan, and Vlassis 2008; Kraemer and Banerjee 2016), allowing agents to utilize global information during training while effectively avoiding the non-stationarity of the environment. However, due to limitations in partial observability, obtaining comprehensive global information is often infeasible. Fortunately, we can replace global information with joint information. Despite this, as the number of agents increases, three challenges remain. Firstly, the use of joint information faces the curse of dimensionality (Oroojlooy and Hajinezhad 2023), which makes it impossible to train in large-scale scenarios, such as the Gather scenario with 495 agents (Terry, Black, and Jayakumar 2020). Secondly, in regions where agents have weak interaction, joint information may not offer significant benefits (Kok and Vlassis 2004), indicating potential information redundancy. Thirdly, communication restrictions may prevent each agent from obtaining joint observations from all other agents, which poses a challenge for CTDE paradigms to train in real-world applications.

With an increasing number of agents, the aforementioned two paradigms may no longer apply to the training of policies. To address the training problem of large-scale multi-agent systems, we consider abstracting the information re-
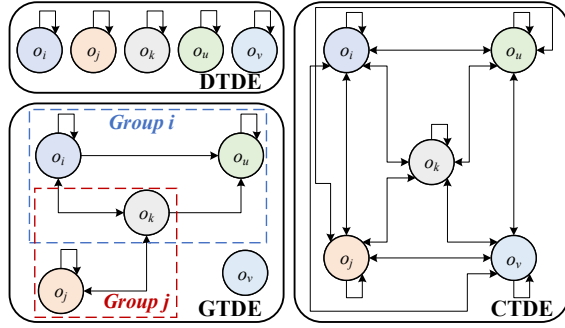
---

*corresponding author

Figure 1: Each circle represents the observation of an agent, and a unidirectional arrow such as $o_i \rightarrow o_k$ indicates that agent i needs the observation of agent k during training.

quired by agents during the training of policies as directed links between agents, thereby obtaining a directed graph. Through this approach, we can obtain that DTDE using individual information[1] represents directed links between agents as self-linked, while CTDE using joint information represents directed links between agents as pairwise links. There is a situation between these two frameworks, as shown in Fig.1, which is the partial links between agents. This also indicates a fact that agents should focus on the specific local information they require (Iqbal and Sha 2019), rather than relying on individual or joint information from agents. Using local information for training of policies can effectively alleviate the problems in DTDE and CTDE, and also provide the possibility for large-scale multi-agent training in real-world applications. For the sake of simplicity, we define the information required by an agent as a group of this agent, as described in Def.1.

In this paper, we propose a novel framework called Grouped Training with Decentralized Execution (GTDE). GTDE mainly considers dividing a large-scale multi-agent system into numerous small groups during training at each timestep, and each agent only needs to utilize the information within the group. However, the dynamic nature of the environment presents an issue in that the information required by each agent can vary over time. Therefore, we propose an adaptive grouping module, in which the number of groups and the number of members within the same group can vary continuously over time. At each timestep, adaptive grouping is only based on observation history to obtain the multivariate Bernoulli distribution of each agent link, and Gumbel-Sigmoid (a variant of Gumbel-Softmax) is used to sample the distribution. After that, given the uncertainty regarding the number of members within the group, we use two methods to implement the group information aggregation module to efficiently consolidate information within the group. During the execution of policies, agents take actions only based on their individual information, such as DTDE

---

[1]We utilize "individual information" to denote the information of an agent, "joint information" to represent the information of all agents, and "local information" to refer to the information of partial agents.

and CTDE. Because value-based methods require the same information to be provided for a single Q-value during training and execution, GTDE primarily focuses on extending multi-agent policy gradient methods within its framework, i.e. actor-critic algorithm. Our contributions are as follows:

- We propose a framework of GTDE oriented towards information requirements for partially observable large-scale multi-agent systems, which focuses on the training of agents from local information rather than global and joint (CTDE) information or individual (DTDE) information.

- We present an adaptive grouping method that combines Gumbel-Sigmoid to address the dynamic information requirements of agents and use group information aggregation to consolidate information within the group.

- We evaluate GTDE on the StarCraft Multi-Agent Challenge V2 (SMACv2) (Ellis et al. 2022) with 20 agents, the Battle scenario with 64 agents, and the Gather scenario with 495 agents (Zheng et al. 2018). Experiments have demonstrated that with an increasing number of agents, GTDE outperforms both DTDE and CTDE.

2 Related Works

Several algorithms (Lowe et al. 2017; Papoudakis et al. 2021; de Witt et al. 2020; Yu et al. 2022) based on CTDE and DTDE encounter challenges such as environmental non-stationarity and the curse of dimensionality, as mentioned in Sec.1. MAAC (Iqbal and Sha 2019) and G2Anet (Liu et al. 2020) identified these issues and addressed them by introducing a soft/hard attention mechanism, which can effectively address the challenges of dimensionality and information redundancy in the centralized critic. However, this approach introduces a new issue. Centralized attention mechanisms necessitate the utilization of information from all agents, making it impossible to train in large-scale scenarios. Therefore, we consider solving the above problem by grouping without introducing other centralized modules.

Some early works utilized the prior domain knowledge to group the agents (Odell, Parunak, and Fleischer 2002; Lhaksmana, Murakami, and Ishida 2013). Mean-Field (Yang et al. 2018) RL uses the actions of nearby agents to estimate the Q-function, but this method does not explicitly define how to determine the neighbors of agents. Subsequently, role-based (Wang et al. 2020; Liu et al. 2022) and subtask-based (Yang et al. 2022; Yuan et al. 2022; Huang et al. 2022) methods group agents to different roles or tasks, and agents of the same role or subtask are naturally grouped together. However, subtask-based and role-based methods require the introduction of an additional centralized module for grouping, and some of them need to perform a clustering algorithm at each time step before running the MARL algorithm, incurring a high time cost. Other works, such as REFIL (Iqbal et al. 2021) divides agents into two groups, VAST (Phan et al. 2021) uses meta-policy for agent grouping, DHCG (Liu et al. 2023) learning grouping policy using RL method, and CommFormer (Hu et al. 2024) learning a static graph. The above grouping algorithms all have the same problem, which is that the number of groups or the

number of members within the same group is fixed, which means that the dynamic nature of the environment has not been fully considered. As the number of agents increases, they need to specify specific grouping numbers for different environments, which is not flexible.

In our method, observation history from a single agent is used to determine the group, and training is conducted end-to-end. Allow agents to learn the number of groups and the number of members within a group by themselves rather than artificially specifying. The model of sharing information during the execution of policies is complementary to GTDE and can combine the two approaches to generate further performance advantages.

3 Background

Decentralized Partially Observable Markov Decision Processes (DEC-POMDP)

We study DEC-POMDP (Oliehoek and Amato 2016), which is defined by a tuple $\langle n, S, Z, O, A, R, P, \gamma \rangle$, where n is the number of agents. $s \in S$ is the true state of the environment. At every stage t, the environment obtains joint observations $z \in Z$ through the observation function $O(s^t, a^{t-1})$ and emits them to each agent. Each agent i has an observation history $\tau_i^t \equiv (o_i^1, \ldots, o_i^t)^2$. Each agent i selects action $a_i \in A_i$ based on a parameterized policy $\pi_i(a_i|\tau_i; \theta)$. The environment transitions to the next state based on the joint actions $a \in A$ of all agents and state transition functions $P(s^{t+1}|s^t, a^t)$, giving the same reward $r^{t+1} \in R(s^t, a^t)$ to all agents through the reward function $R(s^t, a^t)$. The objective of the agents is to maximize the expected return $J = \sum_{t=1}^{T} \gamma^{t-1} r^t$ where $\gamma \in [0, 1)$ is a discount factor and T is the episode length.

Actor-Critic Algorithms

The Policy Gradient (PG) method (Sutton and Barto 2018; Sutton et al. 1999) directly learns parameterized policies to maximize the expected return of the agent, with the optimization objective $J(\theta) = \mathbb{E}_{s \sim p_\pi, a \sim \pi(\theta)}[R(s, a)]$. The gradient with respect to the policy parameters is denoted as

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p_\pi, a \sim \pi(\theta)}[\nabla_\theta \log \pi(a|\tau; \theta) Q_\pi(\tau, a; \varphi)]$$

where p_π is the state distribution under policy π. Actor-Critic Algorithms (Konda and Tsitsiklis 1999) approximates state-action value $Q_\pi(s, a)$ through TD learning. It also uses parameterized functions for updates

$$\mathcal{L}_{TD}(\varphi) = \mathbb{E}_{\mathcal{D}}[(Q^t - (R(s^t, a^t) + \gamma Q^{t+1}))^2] \quad (1)$$

where $\mathcal{D}$ is a set of tuples $(s^t, a^t, r^{t+1}, s^{t+1})$ generated through interaction with the environment and $Q^t = Q(\tau^t, a^t; \varphi)$.

Multi-Agent Actor-Critic Algorithms

In the framework of CTDE, the multi-agent actor-critic algorithm utilizes the joint information of each agent to learn a critic for each agent or directly learn a centralized critic.

---

[2] We use o_i to denote the observation of the i-th agent and o^t to denote the observation at the t-th timestep.

The gradient of the expected return $J(\theta_i)$ with respect to the policy parameters is approximated by

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathcal{D}}[\nabla_{\theta_i} \log \pi(a_i|\tau_i; \theta_i) Q_\pi(\tau, a; \varphi_i)]$$

In the framework of DTDE, the multi-agent actor-critic algorithm only utilizes individual information of each agent to learn a critic for each agent

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathcal{D}}[\nabla_{\theta_i} \log \pi(a_i|\tau_i; \theta_i) Q_\pi(\tau_i, a_i; \varphi_i)]$$

This paper also uses the PPO algorithm and therefore provides the PPO algorithm of DTDE and CTDE paradigms. The policy is updated to maximize

$$J(\theta_i) = \mathbb{E}_{\mathcal{D}}[\min(\frac{\pi_{new}}{\pi} A_\pi, clip(\frac{\pi_{new}}{\pi}, 1 \pm \epsilon) A_\pi)] \quad (2)$$

where $clip(; 1 \pm \epsilon)$ operator clips the input to $1 - \epsilon / 1 + \epsilon$ if it is below/above this value, and $\pi = \pi(a_i|\tau_i; \theta_i)$. For the IPPO algorithm, $A_\pi = A_\pi(\tau_i, a_i; \varphi_i)$ and for the MAPPO algorithm, $A_\pi = A_\pi(\tau, a; \varphi_i)$.

4 Methods

We abstract the information required between agents into a directed graph $G^t = (V^t, E^t)$, where G^t represents the directed graph at timestep t, V^t is the set of nodes representing the agents and E^t is the set of edges between nodes at timestep t representing the information required by the agent. It can be seen that due to the dynamism of multi-agent systems, their directed graphs are inconsistent at each timestep.

Definition 1. *For the i-th node v_i in a directed graph, we set the group of v_i to $g(v_i)$, then*

$$g(v_i) = \{c | I_{E^t}(<v_i, c>) = 1, c \in V^t\} \quad (3)$$

$$\text{where } I_{E^t}(<v_i, c>) = \begin{cases} 1, <v_i, c> \in E^t \\ 0, <v_i, c> \notin E^t \end{cases}, \text{ and } <$$

$u, v >$ represents the directed edge from node u to node v.

GTDE

As shown in Fig.2, the GTDE framework extends the DTDE and CTDE framework by introducing two additional components: the adaptive grouping module and the group information aggregation module. The adaptive grouping module enables the division of agents into distinct groups based on their individual observation history. The group information aggregation module fuses information from group members. Different from the previous grouping method, GTDE does not enforce a fixed number of groups or group members. Instead, it dynamically forms groups based on the observation history of each agent.

CTDE utilizes the joint information of the agent to estimate the value function, while DTDE relies on the individual information of the agent for the same purpose. In contrast, as shown in Fig.1, GTDE employs partial information from the agent within the group to estimate the value function. We observed that when a multi-agent system with n agents is represented as a directed graph, where nodes signify the observations of the agents and edges denote information required between agents. Notably, during the training of policies, the CTDE framework treats the entire multi-agent system as one group, where each group consists of n
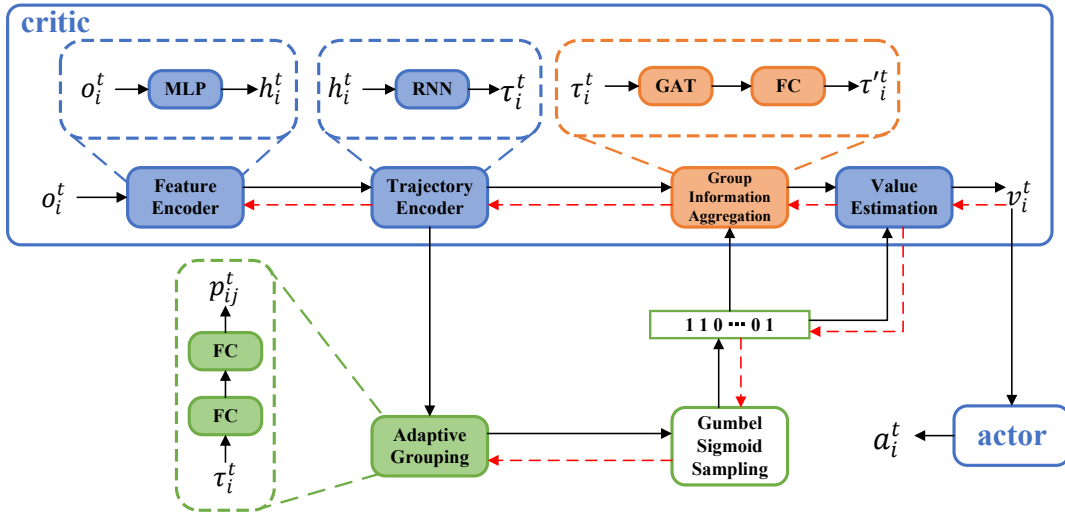
Figure 2: Overview of GTDE framework. The red dashed line represents gradient flow.

members. In contrast, the DTDE framework views the multi-agent system as n separate groups, assigning only one member to each group. GTDE framework takes an intermediate approach, categorizing agents into g groups, with m_g members in each group, where $1 \leq g, m_g \leq n$. CTDE aligns with GTDE when all agents are pairwise linked, while DTDE is equivalent to GTDE when all agents are self-linked. One of the goals of GTDE is to predict the edges of links. Once the link is determined, the system is naturally divided into many groups. GTDE relies on intra-group information during training, rather than global or joint information, and only relies on individual information during execution, making it practical in real-world applications.

Adaptive Grouping

The adaptive grouping module determines the links of agent i through $p_{ij}(c_i = j|\tau_i)$, where τ_i is the observation history of the agent i and $c_i = j$ is the i-th agent is linked to the j-th agent. The agent only needs information from the other agents it is linked to, which avoids the challenge of assigning the agent to a specific group when the number of groups is uncertain. This probability distribution $p_{ij}(c_i = j|\tau_i)$ can be determined using physical distance, but introduces a new problem. Consider an environment with two buttons placed far apart, where pressing them simultaneously yields rewards. In this setting, with many agents present, only the two agents closest to the buttons are grouped together as the most appropriate choice. However, relying solely on physical distance assigns a very small probability of linking them. This also indicates that multi-agent grouping cannot be equated with data clustering. Even agents with highly unrelated features may belong to the same group.

The adaptive grouping module predicts the edge set E^t of the directed graph and uses this edge set to group multi-agent systems using Eq.3. Compared to prior-based fixed grouping methods, the adaptive grouping module dynamically adjusts the number of groups and members within the group at each timestep based on the dynamic changes in the environment. Meanwhile, due to the absence of centralized modules, each agent selects one or more links based on their own observation history during training. After all agents have completed grouping, a multi-agent system is modeled as a directed graph. Specifically, the adaptive grouping module is parameterized by a neural network, with the input being the observation history of a single agent and the output being the n-variate Bernoulli distribution, where n is the number of all agents. Sample on the n-variable Bernoulli distribution to obtain the final determined grouping distribution. It should be noted that each row in the adjacency matrix A^t of a directed graph G^t represents the grouping distribution of an agent. According to the definition of the group, when the grouping distribution of agents is consistent, their belonging groups are also consistent, which makes the number of groups not fixed. At the same time, sampling from the n-variate Bernoulli distribution can obtain different grouping distribution, which makes the number of members in the same group not fixed. Due to the non-differentialable of the sampling operator in backpropagation, we need to use the reparameterization trick of discrete distributions. We can use Gumbel-Sigmoid to efficiently sample n-variable Bernoulli distribution, which is equivalent to Gumbel-Softmax. Please refer to Appendix A for a detailed discussion. As depicted in Sec.3, the $J(\theta)$ under the GTDE framework in the Actor-Critic algorithm is updated to maximize

$$J(\theta) = \mathbb{E}_{\mathcal{D}}[\log \pi(a_i|\tau_i; \theta_i) Q_\pi(\tau'_{g(v_i)}, a_i; \varphi)]$$

where $\tau'_{g(v_i)}$ is the aggregation information of observation history from agent i and its links (To be mentioned in the next section). For the PPO algorithm, consistent with Eq.2, where $A_\pi = A_\pi(\tau'_{g(v_i)}, a_i; \varphi)$. For simplicity, this paper employs parameter sharing, setting $\theta = \theta_1 = \theta_2 = \cdots = \theta_n$.

Group Information Aggregation

After the adaptive grouping module obtains the adjacency matrix of the directed graph G^t, the group information aggregation module can be used to merge the information of the linked agents. Each agent requires its own information, so we need to add a mask matrix and set the diagonal elements in the adjacency matrix to 1 to ensure that each agent is self-linked. Then, we introduce two methods for aggregating group information.

Matrix Multiplication   We multiply the adjacency matrix with the agent trajectory matrix, denoted as $\tau'_{g(v_i)} = A^t \tau^t$. Since the elements in the adjacency matrix are either 0 or 1, the result matrix $\tau'_{g(v_i)}$ only focuses on the trajectory information of linked agents, disregarding the trajectory information of unlinked agents. This implies that agent i only requires the trajectory information of agent j corresponding to the elements in the i-th row vector of the adjacency matrix that have a value of 1. The i-th row in the result matrix is the sum of the information required by the i-th agent. By this means, agent i no longer requires a centralized module to obtain information about all agents. Instead, it only needs to acquire information from a subset of agents to complete its learning process. The use of an adjacency matrix facilitates tensor computations under parameter sharing. In the case of non-parameter sharing, agent i derives a row vector of grouped distributions from its own observation history and then retrieves the trajectory information of agent j corresponding to the elements in the vector that have a value of 1, to obtain Q_π or A_π.

Graph Attention Network   The i-th row of the result matrix can be written as $[\tau'_{g(v_i)}]_i = \sum_{j \ s.t. \ A_{ij}=1} \tau_j$. For some complex scenarios, we can assign different weights to trajectory information instead of giving all selected trajectories a weight of 1. This can be done by introducing a parameter, α, before the trajectory, i.e. $[\tau'_{g(v_i)}]_i = \sum_{j \ s.t. \ A_{ij}=1} \alpha_{ij} \tau_j$. Graph attention network(GAT) (Veličković et al. 2017) can achieve this goal. Specifically, the attention weight e_{ij} of node i to node j is $e_{ij} = w_1 \tau_i + w_2 \tau_j$. By using the above formula, we can obtain the attention matrix e. It is worth noting that although all agent information is used in calculating the attention matrix, unlinked nodes are masked based on the adjacency matrix. As a result, the attention weights of unlinked agents are irrelevant, meaning that any value in those positions is acceptable. Afterwards, we use the Softmax function to obtain the final attention weights, denoted as α_{ij}. Furthermore, although GTDE can complete training with only local information, there are still some challenges in real-world applications. One of them is that although agent i aims to obtain the trajectory information of agent j, the information may not be reached due to communication issues. To address this, we use a mask matrix to randomly drop values in the adjacency matrix with a probability of 0.1. This approach not only simulates the aforementioned communication issue but also prevents the agent from simultaneously selecting information from all agents, thereby avoiding a regression to the CTDE framework.

5   Experiments

In this section, we compared GTDE with CTDE and DTDE and also evaluated variants of GTDE used for ablation study, represented as GTDE-F with fixed grouping distribution and GTDE-U with uniform grouping distribution, as well as GTDE-A with pairwise links, i.e. GTDE degenerated into CTDE. All algorithms were trained using an NVIDIA GeForce RTX 4090 GPU.

Experimental Setup

SMACv2   SMACv2, like SMAC (Samvelyan et al. 2019), consists of two opposing units, each of our units represented by agents trained using the MARL algorithm. Conversely, enemy units utilize a built-in heuristic algorithm. Our primary objective is to eliminate all enemy units within the specified time frame, so the win rate has become an important evaluation indicator for this environment. In contrast to SMAC, SMACv2 predominantly incorporates random team composition, random initial positions, and aligns the attack range of units with the visible range, thereby enhancing the overall randomness of the environment. In this paper, the version we are utilizing is SC2.4.10. GTDE uses GAT to aggregate intra-group information in this scenario.

Performance

Battle and Gather   Battle is a large-scale mixed cooperative competitive scenario where two armies face off against each other in a grid world. Each army consists of 64 agents, and each agent can either move or attack each turn. The goal for both sides is to eliminate the opposing army as quickly as possible. In Gather, agents earn rewards by consuming food. Each piece of food must be broken down by five "attacks" before it can be absorbed. The amount of food on the map is limited, and once a shortage occurs, agents may begin attacking each other in an effort to monopolize the remaining resources. Agents can kill each other with a single attack, and there are a total of 495 agents on the map. In the two scenarios, the observations of each agent are composed of their own features and a circle centered around themselves. GTDE uses matrix multiplication to aggregate intra-group information in this scenario. For Battle and Gather scenarios, we use zero padding observations of dead agents.

Hyperparameter

GTDE focuses on the actor-critic algorithm, so we select MAAC/MAPPO and IAC/IPPO as representatives of CTDE and DTDE respectively, as our baselines. Compare GTDE based on PPO with MAPPO and IPPO, and compare GTDE based on AC with IAC and MAAC. IAC/IPPO trains the critic using the individual observations of the agent, while MAAC/MAPPO trains a centralized critic using the joint observations of all agents. MAPPO and IPPO are used to evaluate SMACv2, while MAAC and IAC are used to evaluate Battle and Gather. For a fair comparison, all basic hyperparameters are set to be consistent. For SMACv2, Battle, and Gather, we trained 10M, 2000, and 1200 environment steps, respectively. For each scenario, we use 5 different random seeds for all algorithms. Additional detailed hyperparameters are addressed in Appendix B.
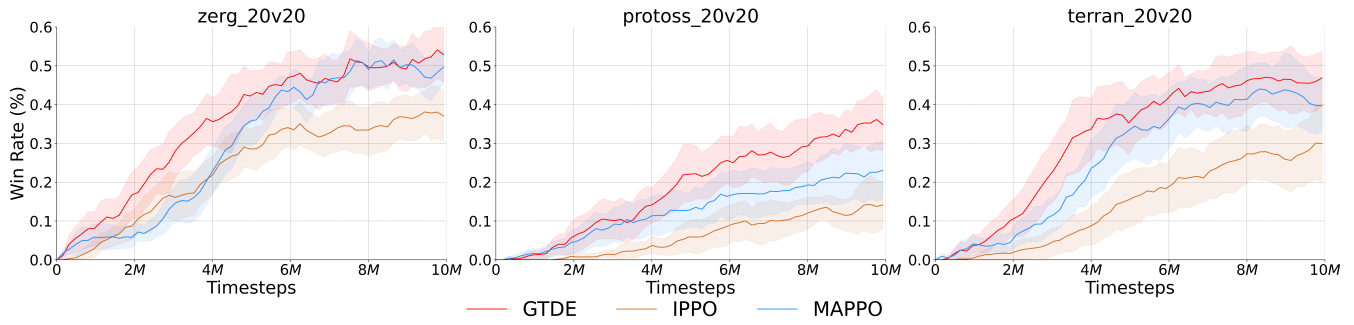
Figure 3: Test curve of average win rate. The shaded area represents the range between the minimum and maximum values across 5 seeds, while the solid line in the center denotes the average value.

	zerg	terran	protoss	Battle
GTDE	**53.4**(3.2)	**48.4**(2.6)	**38.8**(2.5)	52.3(2.8)
CTDE	**50.6**(1.3)	**45.3**(2.3)	26.6(5.7)	**100.0**(0.0)
DTDE	40.8(1.8)	31.3(2.8)	17.6(5.0)	**100.0**(0.0)
GTDE-F	**49.3**(4.1)	42.2(3.5)	32.5(3.7)	97.2(2.6)
GTDE-U	46.4(3.5)	43.5(2.1)	32.8(2.5)	91.8(9.9)
GTDE-A	**50.3**(9.6)	**46.5**(0.9)	**39.1**(4.0)	94.5(8.3)

Table 1: The average win rate and standard deviation were calculated for SMACv2 maps across various models. For the Battle scenario, the win rate represents the outcome of GTDE competing against other algorithms. Values that fall within one standard deviation of the maximum average winning rate are highlighted in bold.

SMACv2 In the SMACv2 benchmark, since the performances of all algorithms in the 5v5 and 10v10 scenarios are similar, we only report the performance results for the 20v20 scenario. Fig.3 shows the test curve of the average win rate. It can be seen that IPPO, which only uses local information, performs the worst in all scenarios. MAPPO and GTDE achieved similar performance in Zerg and Terran scenarios, and better results in the Protoss scenario. This may be due to the lower unit health in Terran and Zerg maps, which leads to shorter episode lengths. In contrast, the presence of shields in the Protoss map can extend the episode duration, giving the agent more time to learn how to effectively utilize local information. As shown in Table 1, we conducted 200 rounds of testing on all models and found that GTDE improved performance by 6.0% and 17.0% compared to CTDE and DTDE.

Battle Since there is no built-in heuristic algorithm for the Battle scenario, we employ self-play for training. Fig.4 shows the total reward curves of GTDE, MAAC, and IAC. It can be seen that IAC experiences significant fluctuations in learning due to relying solely on its individual observation, ultimately converging to a local optimal solution where neither side attacks. In contrast, MAAC, which requires joint observations of all agents, has a much larger input dimension of $(13 \times 13 \times 5 + 32) \times 64$, leading to even lower performance
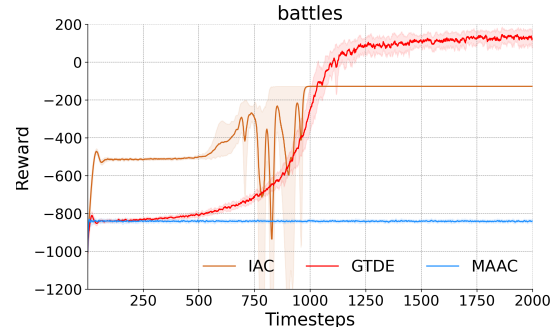

Figure 4: The total reward curve of the Battle scenario(64 v.s. 64).

compared to IAC when using the same network size. GTDE, by utilizing local observations instead of individual or joint ones, achieves a more stable improvement in learning performance. Table 1 shows the win rate of GTDE against IAC and MAAC algorithms, indicating that GTDE defeated them with a 100% win rate. Fig.5 illustrates the links of partial agents in the Battle scenario. It is evident that Agent 15 is positioned far from the other agents, requiring information to encircle the enemy. Similarly, Agent 13, as part of the besieging group, also depends on information from the other besiegers.

Gather In the gather scenario, the large dimensionality of joint observations demands excessive GPU memory, making it impossible to train MAAC in this scenario. Therefore, Fig.6 only shows the total reward curves of GTDE and IAC. It can be seen that the performance gap between GTDE and IAC has further widened. As the scale of agents increases significantly, centralized training methods become impractical, leaving decentralized training as the only option. However, decentralized methods tend to be less effective. GTDE, which leverages local information, offers a better solution to this issue.

Furthermore, Table 2 shows the average agent information used by the three training paradigms in each scenario. Since agents might select agents that have already been eliminated, and these agents do not carry any information (their observa-
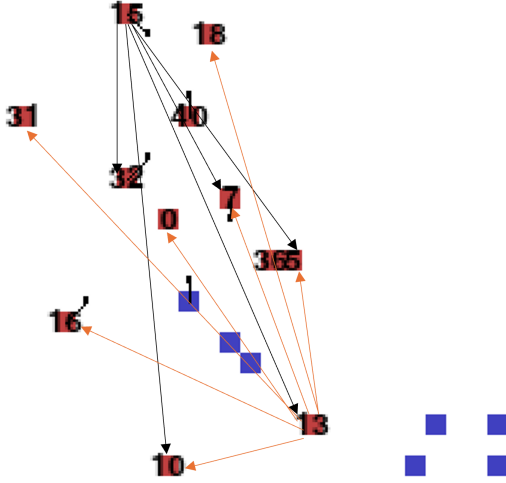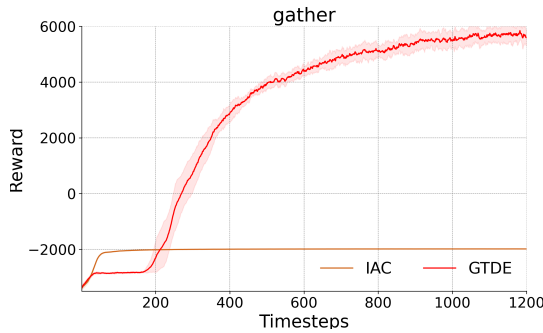
Figure 5: Partial links in the Battle scenario.

Figure 6: The total reward curve of the Gather scenario(495 agents).

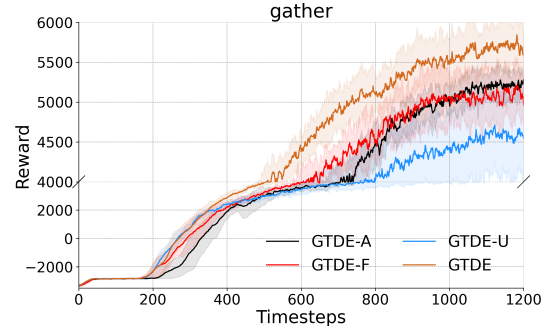| Metric	Scenario	GTDE	CTDE	DTDE
Average node	20v20	$4.9_{(3.0)}$	20	1
information	Battle	$12.0_{(2.0)}$	64	1
$(\lvert g(v_i)\rvert)$	Gather	$103.9_{(2.8)}$	495	1

Table 2: Agent average information required.

Figure 7: Ablations Study of the Gather scenario.

and 10.1% more total rewards than GTDE-U/A/F. Notably, the total rewards of GTDE-F and GTDEA are similar, indicating that using fixed grouping agents can achieve performance comparable to joint observations, which implies that there is information redundancy in joint observations.

6 Conclusion and Future Directions

This paper proposes a GTDE framework to address the limitations of CTDE and DTDE. As the number of agents increases, the performance of DTDE and CTDE significantly deteriorates, with CTDE becoming untrainable due to the high input dimensionality. GTDE is proposed as a solution to this problem. GTDE leverages only the local observations to update policy parameters, making it more suitable for multi-agent systems with a larger number of agents. However, it also faces a challenge in terms of scalability. Unlike CTDE, which uses joint observations and encounters scalability issues due to increasing input dimensions with the number of agents, GTDE avoids this problem but faces a challenge related to the non-scalability of adjacency matrices. It means that new agents can only be linked to existing agents during training and cannot be linked to each other. In the future, we will address or alleviate scalability and train using real-world applications instead of simulators.

References

Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Blei, D. M.; and Frazier, P. I. 2011. Distance Dependent Chinese Restaurant Processes. *J. Mach. Learn. Res.*, 12: 2461–2488.

de Witt, C. S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P. H. S.; Sun, M.; and Whiteson, S. 2020. Is In-

tion values are all zeros), we exclude dead agents from the table. It can be seen that in three scenarios, GTDE, which uses local information, can achieve similar or better performance compared to models using individual or global information. The average information required for GTDE during training has been reduced by five times, making its training cost in real-world applications much lower than that of CTDE.

Ablations Study

We evaluated variants of GTDE to show the effect of the adaptive grouping module. Table 1 shows that GTDE outperforms both GTDE-U with random grouping and GTDE-F with fixed grouping, indicating that GTDE learns the correct grouping distribution. In the SMACv2 benchmark, GTDE using local information can achieve performance similar to GTDE-A with pairwise links. In the Battle scenario, all models of GTDE and GTDE-U/A/F underwent 200 rounds of validation and achieved a win rate of over 90%. Fig.7 shows the broken axis plot of the total reward in the Gather scenario. We can observe that GTDE received 18.2%, 11.6%,

dependent Learning All You Need in the StarCraft Multi-Agent Challenge? *CoRR*, abs/2011.09533.

Dinneweth, J.; Boubezoul, A.; Mandiau, R.; and Espié, S. 2022. Multi-agent reinforcement learning for autonomous vehicles: a survey. *Auton. Intell. Syst.*, 2(1): 27.

Ellis, B.; Moalla, S.; Samvelyan, M.; Sun, M.; Mahajan, A.; Foerster, J. N.; and Whiteson, S. 2022. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning.

Hu, S.; Shen, L.; Zhang, Y.; and Tao, D. 2024. Learning Multi-Agent Communication from Graph Modeling Perspective. In *The Twelfth International Conference on Learning Representations*.

Huang, W.; Li, K.; Shao, K.; Zhou, T.; Taylor, M.; Luo, J.; Wang, D.; Mao, H.; Hao, J.; Wang, J.; et al. 2022. Multiagent q-learning with sub-team coordination. *Advances in Neural Information Processing Systems*, 35: 29427–29439.

Iqbal, S.; de Witt, C. A. S.; Peng, B.; Boehmer, W.; Whiteson, S.; and Sha, F. 2021. Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 4596–4606. PMLR.

Iqbal, S.; and Sha, F. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 2961–2970. PMLR.

Kok, J. R.; and Vlassis, N. 2004. Sparse cooperative Q-learning. In Brodley, C. E., ed., *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM.

Konda, V. R.; and Tsitsiklis, J. N. 1999. Actor-Critic Algorithms. In Solla, S. A.; Leen, T. K.; and Müller, K., eds., *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, 1008–1014. The MIT Press.

Kraemer, L.; and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190: 82–94.

Krnjaic, A.; Thomas, J. D.; Papoudakis, G.; Schäfer, L.; Börsting, P.; and Albrecht, S. V. 2022. Scalable Multi-Agent Reinforcement Learning for Warehouse Logistics with Robotic and Human Co-Workers. *CoRR*, abs/2212.11498.

Laurent, G. J.; Matignon, L.; and Fort-Piat, N. L. 2011. The world of independent learners is not markovian. *Int. J. Knowl. Based Intell. Eng. Syst.*, 15(1): 55–64.

Lhaksmana, K. M.; Murakami, Y.; and Ishida, T. 2013. Role Modeling for Adaptive Multiagent Systems Engineering. In *2013 IEEE/WIC/ACM International Conferences on Intelligent Agent Technology, IAT 2013, 17-20 November 2013, Atlanta, Georgia, USA*, 287–292. IEEE Computer Society.

Liu, Y.; Li, Y.; Xu, X.; Liu, D.; and Dou, Y. 2022. ROGC: Role-Oriented Graph Convolution Based Multi-Agent Reinforcement Learning. In *IEEE International Conference on Multimedia and Expo, ICME 2022, Taipei, Taiwan, July 18-22, 2022*, 1–6. IEEE.

Liu, Y.; Wang, W.; Hu, Y.; Hao, J.; Chen, X.; and Gao, Y. 2020. Multi-Agent Game Abstraction via Graph Attention Neural Network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 7211–7218. AAAI Press.

Liu, Z.; Wan, L.; Sui, X.; Chen, Z.; Sun, K.; and Lan, X. 2023. Deep Hierarchical Communication Graph in Multi-Agent Reinforcement Learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, 208–216. ijcai.org.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 6379–6390.

Mguni, D. H.; Chen, H.; Jafferjee, T.; Wang, J.; Yue, L.; Feng, X.; McAleer, S. M.; Tong, F.; Wang, J.; and Yang, Y. 2023. MANSA: Learning Fast and Slow in Multi-Agent Systems. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 24631–24658. PMLR.

Odell, J.; Parunak, H. V. D.; and Fleischer, M. 2002. The Role of Roles in Designing Effective Agent Organizations. In Garcia, A. F.; de Lucena, C. J. P.; Zambonelli, F.; Omicini, A.; and Castro, J., eds., *Software Engineering for Large-Scale Multi-Agent Systems, Research Issues and Practical Applications [the book is a result of SELMAS 2002]*, volume 2603 of *Lecture Notes in Computer Science*, 27–38. Springer.

Oliehoek, F. A.; and Amato, C. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer. ISBN 978-3-319-28927-4.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. 2008. Optimal and Approximate Q-value Functions for Decentralized POMDPs. *J. Artif. Intell. Res.*, 32: 289–353.

Oroojlooy, A.; and Hajinezhad, D. 2023. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.*, 53(11): 13677–13722.

Papoudakis, G.; Christianos, F.; Schäfer, L.; and Albrecht, S. V. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In Vanschoren, J.; and Yeung, S., eds., *Proceedings of the Neural Information Processing Systems Track on Datasets and*

Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.

Phan, T.; Ritz, F.; Belzner, L.; Altmann, P.; Gabor, T.; and Linnhoff-Popien, C. 2021. VAST: Value Function Factorization with Variable Agent Sub-Teams. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 24018–24032.

Ruan, J.; Hao, X.; Li, D.; and Mao, H. 2023. Learning to collaborate by grouping: A consensus-oriented strategy for multi-agent reinforcement learning. In *ECAI 2023*, 2010–2017. IOS Press.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *SCIENCE*, 362(6419): 1140+.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction(2nd edition)*. MIT Press.

Sutton, R. S.; McAllester, D. A.; Singh, S.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Solla, S. A.; Leen, T. K.; and Müller, K., eds., *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, 1057–1063. The MIT Press.

Terry, J. K.; Black, B.; and Jayakumar, M. 2020. MAgent. https://github.com/Farama-Foundation/MAgent. GitHub repository.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wunsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *NATURE*, 575(7782): 350+.

Wang, Q.; He, C.; Jaffrès-Runser, K.; Huang, J.; and Xu, Y. 2022. Timely-throughput optimal scheduling for wireless flows with deep reinforcement learning. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 1–11. IEEE.

Wang, Q.; Huang, J.; and Xu, Y. 2024. Scheduling of Real-Time Wireless Flows: A Comparative Study of Centralized and Decentralized Reinforcement Learning Approaches. *IEEE/ACM Transactions on Networking*, 32(5): 3699–3714.

Wang, T.; Dong, H.; Lesser, V. R.; and Zhang, C. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 9876–9886. PMLR.

Yang, M.; Zhao, J.; Hu, X.; Zhou, W.; Zhu, J.; and Li, H. 2022. LDSA: Learning Dynamic Subtask Assignment in Cooperative Multi-Agent Reinforcement Learning. In *NeurIPS*.

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean Field Multi-Agent Reinforcement Learning. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 5567–5576. PMLR.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yuan, L.; Wang, C.; Wang, J.; Zhang, F.; Chen, F.; Guan, C.; Zhang, Z.; Zhang, C.; and Yu, Y. 2022. Multi-Agent Concentrative Coordination with Decentralized Task Representation. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 599–605. ijcai.org.

Zheng, L.; Yang, J.; Cai, H.; Zhou, M.; Zhang, W.; Wang, J.; and Yu, Y. 2018. MAgent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhou, W.; Chen, D.; Yan, J.; Li, Z.; Yin, H.; and Ge, W. 2022. Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Auton. Intell. Syst.*, 2(1).