GCS: Graph-Based Coordination Strategy for Multi-Agent Reinforcement Learning

Jingqing Ruan[12], Yali Du*[3], Xuantang Xiong[1], Dengpeng Xing[1], Xiyun Li[1], Linghui Meng[1], Haifeng Zhang[1], Jun Wang[4], Bo Xu*[1]
[1]Institute of Automation,Chinese Academy of Sciences,Beijing,China
[2]School of Future Technology,University of Chinese Academy of Sciences,Beijing,China
[3]King's College London, [4]University College London
{ruanjingqing2019,xiongxuantang2021,lixiyun2020,dengpeng.xing,menglinghui2019,haifeng.zhang,xubo}@ia.ac.cn
yali.du@kcl.ac.uk,jun.wang@cs.ucl.ac.uk

ABSTRACT

Many real-world scenarios involve teams of agents who have to coordinate their policies to achieve a shared goal. Most previous studies mainly focus on learning decentralized control policies to maximize a common reward and rarely consider the coordinated control policies, which is critical in dynamic, complicated environments. Out of the fully decentralized rules, we propose a graph-based coordination strategy (GCS) which consists of a graph generator and a factored coordination policy. We adopt an encoder-decoder model as the generator to construct a directed acyclic graph (DAG), dynamically representing the underlying decision structure. We also apply the DAGness and depth constrained optimization in the graph generator to balance efficiency and performance. To exploit the underlying decision structure, we investigate the factored coordination policy to obtain the optimally coordinated behaviors among agents. We train the graph generator and the factored coordination policy simultaneously to maximize the discounted return. Empirical evaluations on Collaborative Gaussian Squeeze, Cooperative Navigation, and Google Research Football demonstrate the superiority of the proposed method.

KEYWORDS

Action Coordination Graph, Multi-Agent Systems, Reinforcement Learning

ACM Reference Format:
Jingqing Ruan[12], Yali Du*[3], Xuantang Xiong[1], Dengpeng Xing[1], Xiyun Li[1], Linghui Meng[1], Haifeng Zhang[1], Jun Wang[4], Bo Xu*[1]. 2022. GCS: Graph-Based Coordination Strategy for Multi-Agent Reinforcement Learning. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 11 pages.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has shown exceptional results in many real-life applications, such as multiplayer games [1, 2], traffic control [3], and social dilemmas [4]. A suitable control policy is extremely important in multi-agent systems (MAS). One choice is the centralized control policy [5], whose major constraint is the poor scalability on high-dimensional state and action spaces. Another is the decentralized control policy [2, 6–10] allowing agents to make decisions independently, but it is difficult to obtain coordinated behaviors. We give a traffic example for intuitive understanding, where there is a traffic intersection without traffic lights and where many vehicles are blocked at the multiple-lane crossroad. It leads to a high probability of being congested if all vehicles take actions simultaneously without a rational sequence. This dilemma may be properly solved if those vehicles move orderly based on a local action-coordinated structure. This example reflects that it is imperative to improve the fully decentralized decision-making process. Therefore, a solution to alleviate the above issues is to develop a coordinated control policy to obtain optimal behaviors among agents.

Several approaches are reported to solve the problem of action coordination. BiAC [11] mainly focuses on the coordination for the asynchronous decision of two agents. The multi-agent rollout algorithm [12] provides a theoretical view of executing a local rollout with some coordinating information, but is limited to the agent-by-agent decision dependency structure. Although these works investigate the action execution order of two-agent and agent-by-agent, it is still insufficient to characterize all agents' complicated and dynamic underlying decision dependency structure. Moreover, we remark that representing the underlying decision dependency structure and then controlling the action execution is essential to improve coordination.

In this work, we propose a graph-based coordination strategy called GCS that learns coordinated behaviors through factorizing the coordinated strategy into graph generator and factored coordination policy. The former aims to learn an action coordination graph (ACG) that properly represents the decision dependency. The latter further coordinates the dependent behaviors among agents exploiting the underlying decision dependency. We adopt the simultaneous training of the graph generator and the factored coordination policy to maximize the discounted return. As for ACG, we employ directed acyclic graphs (DAGs), whose nodes represent agents and whose directed edges denote action dependency of the associated agents. Moreover, we propose the DAGness and DAG depth constrained optimization in the graph generator to balance efficiency and performance.

The contributions of this paper can be summarized as follows.

- As far as we know, we are the first to introduce the directed acyclic graphs to action coordination, dynamically representing the underlying decision dependency for MAS.

∗ Corresponding author.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

- We propose a DAGness and DAG depth constrained optimization in the graph generator, achieving a trade-off between decision-making efficiency and performance.
- Empirical evaluations on several challenging MARL benchmarks (Collaborative Gaussian Squeeze, Cooperative Navigation, and Google Football) show that our method can achieve superior performance and obtain meaningful results consistently with intuitive expectations.

2 RELATED WORK

Deep reinforcement learning has been applied to address complex decision problems successfully [13–15]. Due to the widespread existence of multi-agent tasks, MARL has attracted more and more attention. Learning appropriate control policies is important to obtain the maximum cumulative discount return. Based on the structure of execution schemes, we classify the existing approach into three categories.

First, a fully independent execution scheme allows agents to determine actions according to their individual policy without any interaction. A line of approaches focuses on the value-based methods, such as IQL [16], VDN [6], QMIX [7] and QTRAN [9], which assign each agent an independent policy for execution. Another line of researches including MAAC [5], COMA [17], and LIIR [10] extends the actor-critic algorithm [18] to the multi-agent case, where each actor represent an individual policy for each agent.

The communication-based independent execution scheme is widely used, which allows using extensive information in its individual decision making [19]. In this scheme, agents learn how to transmit informative messages and process the messages during training. Then agents exchange the messages to determine their actions individually during independent execution. Representative methods [20–25] autonomously learn communication protocols that are required in generating informative messages to determine whom to communicate with and what messages to transmit for assisted decision-making.

The coordinated execution scheme, where agents develop their policies conditioned on other agents' actions and make decisions in a coordinated manner, is important in MAS. There are some methods that implicitly model the coordinated behaviours from the perspective of coordination graph. DCG [26] uses pairwise graph to propagate belief for joint decision while DICG [27] focuses on generating the coordination graph with soft edge weights for message passing. DGN [28] uses graph attention network as embedding extractor to assist the decision-making. Some methods are proposed to explicitly model the coordinated behaviours orderly, such as Bi-AC [11], and multi-agent rollout [12], which are proposed to utilize two-agent and agent-by-agent dependency structure respectively to help agents make decisions orderly and promote action coordination. Similar but essentially distinct, we introduce a mechanism to learn the optimal DAG structure which represents the decision dependency among agents.

Moreover, the generation of the DAGs is an essential part of our work. Recently, some continuous optimization approaches [29–32] have been proposed to recover the DAGs through structure learning. The method [33], closely related to our work, uses reinforcement learning as its search strategy to maximize the predefined score function. Borrowed from the idea of [33] that obtains the optimal DAG using reinforcement learning, we construct a graph generator module to generate the DAG structure as the action coordination graph and regard the extrinsic reward as the incentives jointly training with MARL tasks.

3 PROBLEM SETUP

MMDPs . The cooperative multi-agent problems can be modeled as multiagent Markov decision processes (MMDPs) [34], which can be expressed as a tuple $< \{\mathcal{I}\}, \mathcal{S}, \{\mathcal{U}^i\}_{i=1}^N, \mathbb{P}, r, \gamma >$. $i \in \mathcal{I}$ is the i^{th} player, $\mathcal{S}$ is the global state space, and $\mathcal{U}^i$ denote the action space for the i^{th} player. Intuitively, the agent i will select an individual action u^i to perform and execute it. We label $\boldsymbol{u} := (u^1, ..., u^N)$ the joint actions for all players. $\mathbb{P}$ is the transition dynamics, and $\mathbb{P}(\cdot|s, \boldsymbol{u})$ gives the distribution of the next state at state s taking action $\boldsymbol{u}$. All agents share the same reward function $r(s, \boldsymbol{u}) : s \times \boldsymbol{u} \rightarrow R$. $\gamma \in (0, 1)$ denote a discount factor and $\tau = (s_0, \boldsymbol{u}_0, s_1, ...)$ denote the trajectory induced by the policy $\boldsymbol{\pi} = \{\pi^i\}_{i=1}^N$. All the agents coordinate together to maximize the cumulative discounted return $\mathbb{E}_{\boldsymbol{\tau}\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{u}_t)\right]$.

Factored MMDPs . We formalize our problem based on MMDPs as factored MMDPs. Different from MMDPs, where all actions are taken simultaneously and do not depend on each other, we endow the hierarchy order to the joint action based on the learned DAG structure $\mathcal{G}$, called action coordination graph (ACG). The adjacency matrix A representing the ACG denotes the decision dependency from the graph generator ρ. With A, we can define $\boldsymbol{\pi} = \prod_{i=1}^{n} \pi^i(u^i|o^i, u^{pa(i)})$ as the factored coordination policy for the i^{th} player, where $pa(i)$ is the parents of agent i and $u^{pa(i)}$ is the actions taken from the parents, whose order is generated from A. Note that fully decentralized policy is a special case of our factored coordination policy where all the agents have no parents.

Figure 1 gives an example of the DAG and the relationship between nodes. The nodes in $\mathcal{G}$ correspond to the agents in MAS, and the parent-child relationship represents the hierarchical decision dependency among agents. For example, the case that the node D in the graph has two parents $\{B, C\}$ illustrates that the action taken by the agent D is constrained by B and C.

Now, the graph-based coordinated strategy is factorized as:

$$\pi(\boldsymbol{u}, A|s) = \rho(A|s) \prod_{i=1}^{n} \pi^i(u^i|o^i, u^{pa(i)\sim A}), \tag{1}$$

where ρ is the graph generator, and π^i is factored coordination policy.

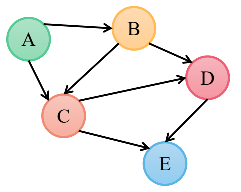
The state-action value function is therefore defined as:

$$Q^{\pi}(s, \boldsymbol{u}^{A\sim\rho}) = \mathbb{E}_{\pi,\rho}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, u_{t+k}) | S_t = s, A_t = \boldsymbol{u}^{A\sim\rho}\right]. \tag{2}$$

4 METHODOLOGY

The overall goal is to maximize the cumulative return, denoted as:

$$\eta = \mathbb{E}_{A\sim\rho, \boldsymbol{u}\sim\pi(\cdot|s,A)}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, \boldsymbol{u}_{t+k})\right]. \tag{3}$$



Directed Acyclic Graph $\mathcal{G}$

Vertex	Parents
A	{}
B	{A}
C	{A,B}
D	{B,C}
E	{C,D}

Relationship between Nodes

Figure 1: The left is the schematic diagram of a directed acyclic graph $\mathcal{G}$ with vertices $|\mathcal{V}| = 5$ and edges $|\mathcal{E}| = 7$. The right is the parent-child relationship about $\mathcal{G}$.

Now we further elaborate on the derivation of the factored coordination policy $\pi_{\theta^i}(u^i|o^i, u^{pa(i)})$ and the graph generator $\rho_\varphi(A|s)$, respectively.

4.1 Factored Coordination Policy

Given a known graph generator ρ (elaborated in Section 4.2), we have $A \sim \rho$ to represent the underlying decision dependency. Based on it, we can factor the decision policy as $\boldsymbol{\pi} = \prod_{i=1}^{N} \pi^i(u^i|o^i, u^{pa(i)\sim A})$, called factored coordination policy. We explore the optimal factored coordination policy $\boldsymbol{\pi}^*$ that obtains the final joint action $\{u^i\}_{i=1}^N$ as follows.

With Equations (1) and (3), we can write the expected return for agent i as:

$$\begin{aligned}\eta^i &= \mathbb{E}_{s\sim p^{\boldsymbol{\pi}},(\boldsymbol{u},A)\sim\pi}\left[\pi(\boldsymbol{u},A|s)Q^i_{\boldsymbol{\pi}}(s,\boldsymbol{u})\right]\\ &= \mathbb{E}_{s\sim p^{\boldsymbol{\pi}},u^i\sim\pi^i,A\sim\rho}\left[\rho(A|s)\prod_{i=1}^{N}\pi^i(u^i|o^i,u^{pa(i)\sim A})Q^i_{\boldsymbol{\pi}}(s,\boldsymbol{u})\right].\end{aligned} \tag{4}$$

The factored coordination policy $\boldsymbol{\pi} = \{\pi_1, \pi_2, ..., \pi_N\}$ for N agents can be parameterized by $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_N\}$. Correspondingly, the gradient of the expected return for agent i is expressed as:

$$\begin{aligned}&\nabla_{\theta^i}\eta(\theta^i) = \nabla_{\theta^i}\eta^i\\ &= \mathbb{E}_{s\sim p^{\boldsymbol{\pi}},u^i\sim\pi^i}\left[\nabla_{\theta^i}\log\pi^i(u^i|o^i,u^{pa(i)})\textstyle\sum_A\rho(A|s)Q^i_{\boldsymbol{\pi}}(s,\boldsymbol{u})\right].\end{aligned} \tag{5}$$

By applying the mini-batch technique to the off-policy training, the gradient can be approximately estimated as:

$$\nabla_{\theta^i}\eta(\theta^i) = \mathbb{E}_{(s,\boldsymbol{o},A,\boldsymbol{u})\sim\mathcal{D}}\left[\nabla_{\theta^i}\log\pi^i(u^i|o^i,u^{pa(i)})Q^i_{\boldsymbol{\pi}}(s,\boldsymbol{u})\right], \tag{6}$$

where $\mathcal{D}$ is the experience replay buffer, recording experiences of all agents. Moreover, the centralized action-value function Q^i_π can be updated as:

$$\mathcal{L}(\phi) = E_{(s,\boldsymbol{o},A,\boldsymbol{u})\sim\mathcal{D}}\left[\left(Q^i_{\boldsymbol{\pi}}(s,\boldsymbol{u}) - y^i\right)^2\right] \tag{7}$$

where $y^i = r + \gamma\max_{\boldsymbol{u}'}Q^i_{\boldsymbol{\pi}^-}(s',\boldsymbol{u}')$ is the learning target and $Q^i_{\boldsymbol{\pi}^-}$ is the target network parameterized by θ^{i^-}.

During the training process, the factored coordination policy $\boldsymbol{\pi} = \{\pi_1, \pi_2, ..., \pi_N\}$ and the graph generator ρ are updated iteratively. We will describe how to find the optimal graph generator $\rho(\cdot)$ under a given policy $\boldsymbol{\pi}$.

4.2 Graph Generator

The graph generator aims to generate the DAG $\mathcal{G}$ to define the decision dependency among agents. We will introduce it in detail from three aspects: (a) DAGness constraint, (b) DAG Depth constraint, and (c) optimization objective.

DAGness constraint. The acyclicity constraint is an important issue in our problem setting. In this work, we also use the penalty terms like [29, 32, 33] to ensure acyclicity. The result in [29] shows that the directed graph $\mathcal{G}$ with binary adjacency matrix A is acyclic if and only if:

$$g(A) := \text{trace}(e^{A\circ A}) - d = 0, \tag{8}$$

where $e^{A\circ A}$ is the matrix exponential, $A \circ A$ guarantees the non-negativity, and d is the number of nodes in the DAGs. The 'trace' of a matrix is defined as the sum of the diagonal elements [29]. The constraint function g should satisfy that: (a) its derivatives are computable, and (b) g can be the measurement of DAGs.

DAG Depth constraint. Moreover, taking the trade-off between efficiency and performance into account, we claim that the maximum depth of graph structure should be adjustable over different tasks. Therefore, we propose an alternative constraint to control the hierarchy of the generated DAGs as follows.

Definition 1. *A square matrix A is a Nilpotent Matrix [35], if*

$$A^k = O \text{ and } A^{k-1} \neq O,\ \exists k \in \mathbb{Z}^+,$$

where O is the zero matrix and A is called the Nilpotent of index k.

Proposition 1. *Let A be an adjacency matrix for a directed acyclic graph, then the maximal length between any two nodes i and j is k if A is Nilpotent of index k.*

We provide a detailed proof of proposition 1 in Appendix A.1. Here, we define $c(A^k) := sum(A^k) = \sum_i\sum_j A^k_{ij} = 0$, which is equivalent to $A^k = O$. We remark that as long as constraint $A^k = O$ holds, it can be guaranteed that the maximal length between any two nodes i and j of the DAG does not exceed k.

Optimization objective. Based on the foregoing, we can optimize ρ parameterized by φ with the maximal length k by:

$$\begin{aligned}&\max\ \eta(\varphi) = \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k r^\rho(s_{t+k},\boldsymbol{u}_{t+k})\right],\\ &s.t.\ g(W) = 0, c(W^k) = 0\end{aligned} \tag{9}$$

where $W = \rho_\varphi(\cdot)$ denotes the weight matrix generated from the graph generator ρ. Then the weight matrix is modeled as a Bernoulli distribution, from which the binary adjacency matrix A is sampled. Here, we use the constraints of the weight matrix $g(W)$ and $c(W^k)$ to approximate those of the adjacency matrix $g(A)$ and $c(A^k)$ due to the consistency of representing the graph structure. With this approximation, we restate η^i as:

$$\eta^i = \mathbb{E}\left[\rho\left(W|s\right)\prod_{i=1}^{N}\pi^i\left(u^i|o^i,u^{pa(i)}\right)Q^i_{\boldsymbol{\pi}}(s,\boldsymbol{u})\right]. \tag{10}$$

Fixing factored coordination policy $\boldsymbol{\pi}$, we approximate the graph generator ρ as follows. We augment the original problem shown

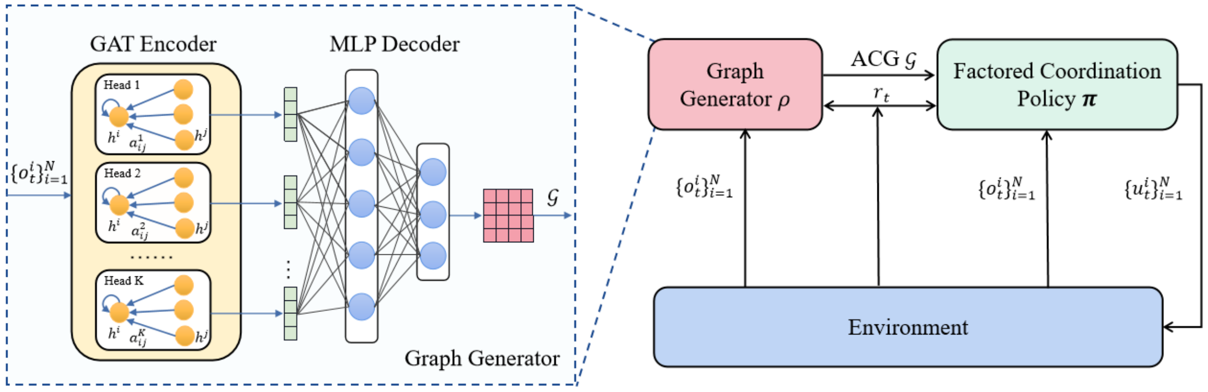


Figure 2: The proposed framework for the interaction process. The left is the graph generator including an encoder-decoder neural network model which is used to generate the DAG from the observed data.

in Equation (9) with a quadratic penalty using the augmented Lagrangian technique [36]:

$$\begin{aligned} \min_{\varphi} \quad & -\eta(\varphi) + \frac{\xi}{2}\left[||g(W)||^2 + ||c(W^k)||^2\right] \\ s.t. \quad & g(W) = 0, c(W^k) = 0 \end{aligned}, \tag{11}$$

with the penalty $\xi > 0$.

Next, we convert the Equation (11) to an unconstrained Lagrangian function:

$$\begin{aligned} L(\varphi, \lambda_1, \lambda_2) = & -\eta(\varphi) + \frac{\xi}{2}\left[||g(W)||^2 + ||c\left(W^k\right)||^2\right] \\ & + \lambda_1 g(W) + \lambda_2 c\left(W^k\right) \end{aligned}. \tag{12}$$

Inspired by [37, 38], the augmented Lagrangian can perform dual gradient ascent to optimize as:

$$\begin{aligned} \varphi^{t+1} &= \arg\min_{\varphi} L(\varphi, \lambda_1^t, \lambda_2^t) \\ \lambda_1^{t+1} &= \lambda_1^t + \xi g(W^{t+1}) \\ \lambda_2^{t+1} &= \lambda_2^t + \xi c\left((W^{t+1})^k\right). \end{aligned} \tag{13}$$

When ξ is large enough, λ_1 and λ_2 converge to the optimal solutions. Moreover, we can set ξ a large number due to the property that it solves a constrained problem by the solutions of unconstrained problems without increasing the penalty parameter ξ to infinity as described in [29, 39]. In our experiments, we set $\xi = 10^{16}$.

Proposition 2. *The gradient for Equation (12) to optimize the coordination graph generation policy can be derived as follows:*

$$\begin{aligned} \nabla_{\varphi} L(\varphi, \lambda_1, \lambda_2) = E_{s\sim p, W\sim\rho_{\varphi}(\cdot|s)} & \Big[\nabla_{\varphi} \log \rho_{\varphi}(W|s) \sum_{u^i} \pi^i(u^i|o^i, \\ & u^{pa(i)}) \cdot Q^i(o^i, u^i) - \lambda_1 (e^{W\circ W})^T \cdot 2W - \\ & \lambda_2 \textstyle\sum_{i,j}\left[kW^k W^{-1}\right]_{ij}\Big], \end{aligned}$$

We provide a detailed proof of proposition 2 in Appendix A.2.

In proposition 2, we remark that after considering the influence of various decision dependencies on the reinforcement learning tasks, we can obtain the optimal underlying graph structure that makes the best response to MARL tasks.

4.3 Implementation Details

As shown in Figure 2, the proposed framework include the factored coordination policy and the graph generator, which will be elaborated below.

Factored Coordination Policy. The factored coordination policy can be obtained from the standard multi-agent actor-critic framework. As for the policy π^i of the actor, we use the RNN network with the stochastic policy gradient to model the action distributions. The critic used to criticize the joint actions made by the actors is a three-layer feed-forward neural network activated by the ReLU units, denoted as $f_{critic}^{'}(o_t, u_t) = ReLU(MLP\langle o_t, u_t\rangle)$.

Graph Generator. As shown in Figure 2, the graph generator $\rho_{\varphi}(\boldsymbol{o})$ adopt an encoder-decoder module used to find the ACG. The GAT-based encoder can model the interplay of agents and extract the further latent representations. The MLP-based decoder is used to recover the pairwise relationship between agents to generate an ACG. The graph generator takes the local observations of the agents as input and outputs the ACG $\mathcal{G}$ to obtain the decision dependency for the decision-making process of the factored coordination policy, elaborated as follows.

The graph generator contains two sub-modules. Firstly, we use the graph attention network (GAT) [40] as the attention-based encoder to extract the latent information for the graph structure generation. First, the simple feature is extracted by a multi-layer perceptron (MLP) as an initial step:

$$\{h_t^i\}_{i=1}^N = f_{MLP}(\{o_t^i\}_{i=1}^N, \{u_{t-1}^i\}_{i=1}^N). \tag{14}$$

Due to the sufficiently expressive power of GAT, we use it to extract the further latent information of the simple feature. We compute the importance coefficients through the attention mechanism:

$$a_t^{ij} = \frac{\exp(h_t^i \boldsymbol{W}(h_t^j \boldsymbol{W})^T/\sqrt{d}))}{\sum_{k\in\{I\}^{-i}} \exp(h_t^i \boldsymbol{W}(h_t^k \boldsymbol{W})^T/\sqrt{d})}, \sum\nolimits_{j\in\{I\}^{-i}} a_t^{ij} = 1, \tag{15}$$

where $\boldsymbol{W} \in R^{d'\times d}$ is a learnable weight matrix, d' and d denote the dimensions of the input vector and the latent vector, respectively, and $k \in \{I\}^{-i}$ indexes other agents except the agent i.

Then the multi-head attention is used to stabilize the learning process of self-attention, and the final latent feature is as follows:

$$h_t^{i'} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{j\in\{I\}^{-i}} a_{t,m}^{ij} \boldsymbol{W}^m h_t^j\right). \tag{16}$$

where M is the number of attention heads. $a_{t,m}^{ij}$ are importance coefficients computed by the k-th attention mechanism, and W^m is the corresponding weight matrix.

Another sub-module in the graph generator is the decoder that generates a weight matrix used to sample the graph structure. Since the GAT-based encoder has already provided sufficiently expressive features among agents, a single-layer decoder is enough to easily construct the pairwise relationship between the encoder outputs to find a better structure of DAG for the optimal policy.

$$f_t^{ij}(\boldsymbol{h}_t^l, \boldsymbol{h}_t^r, u) = \sigma(u^T tanh(\boldsymbol{W_l h}_t^l + \boldsymbol{W_r h}_t^r)), \tag{17}$$

where $\boldsymbol{h}_t^l$ and $\boldsymbol{h}_t^r$ are agents' higher-level representations from two encoder outputs, $\boldsymbol{W_l}, \boldsymbol{W_r} \in \mathbb{R}^{d_h \times d_e}, u \in \mathbb{R}^{d_h \times 1}$ are trainable parameters, and d_h, d_e are the hidden dimension and encoder output dimension, respectively.

Moreover, the logistic sigmoid function $\sigma(\cdot)$ generates the probability for constructing the Bernoulli distribution from which the binary adjacency matrix A is sampled. The binary adjacency matrix forms a directed graph corresponding to the ACG $\mathcal{G}$. Here, we denote this graph generation process as $\mathcal{G} \sim \rho_\varphi(\boldsymbol{h})$.

Parameter Setting. In the graph generator, the attention head in the GAT encoder is set to 8, the stacked attentional layers are set to 4, and the hidden units in the MLP is set to 64. In the factored coordination policy, the actor critic architecture is adopted. The recurrent layer comprised of a GRU with a 64-dimensional hidden state, with a fully-connected layer before and after, is used as the actor. The critic is a two-layer MLP with the ReLu activation.

4.4 Algorithm Description

The main procedures are summarized in Algorithm 1, where $\nabla_\varphi L(\varphi)$, $\eta_{\theta^i}(\theta^i)$, and $\nabla_\phi \mathcal{L}(\phi)$ are optimized. Our ultimate goal is to obtain the optimal factored coordination policy $\{\pi^i\}_{i=1}^n$. The graph generator ρ is an intermediate used to access an excellent graph structure in guiding the decision-making sequence among agents to achieve a high degree of multi-agent coordination. The graph generator is a pluggable module that can be replaced by other algorithms for solving the DAGs. Note that DAGs are necessary because we need an execution structure that can determine a clear sequence of the actions, and therefore it should be directed and not circular. The policy solver is a universal module, which, in general, one can choose from a diverse set of cooperative MARL algorithms [2, 5, 41].

5 EXPERIMENTS

We evaluate the effectiveness of our algorithm on three different environments: Collaborative Gaussian Squeeze[1] [9], Cooperation Navigation[2] [2], and Google Research Football[3] [42].

[1]The MGS environment is at https://github.com/Sonkyunghwan/QTRAN
[2]The code is at https://github.com/openai/multiagent-particle-envs
[3]The code is at https://github.com/google-research/football

Algorithm 1: The optimization of GCS.

```
Ensure factored coordination policy $\pi^i$ and graph
  generator $\rho$;
Initialize $\epsilon, \gamma, M, C$, and the replay buffer $\mathcal{D} \leftarrow \emptyset$;
Initialize the parameters $\theta^i, \phi^i$ for the factored
  coordination policy networks, where $i = 1, ..., n$, and $\varphi$ for
  the graph generator network;
Initialize the target networks $\theta^{i^-} = \theta^i$ and $\phi^{i^-} = \phi^i$;
for each episode do
    Initial state $\leftarrow \{o_0^i\}_{i=1}^n$; // drop $i$ as $\boldsymbol{o_0}$ for clarity;
    Initialize $h_0^{(1)}, ..., h_0^{(n)}$ for RNN states;
    for each timestep $t$ do
        Get $\mathcal{G}_t \sim \rho(\cdot|\boldsymbol{o}_t)$;
        Get the topological order $A_t = f(\mathcal{G}_t)$;
        for $i$ in order $A_t$ do
            augment the observations as
              $\hat{o}_t^i = (o_t^i, u_{t-1}^i, u_t^{pa(i)})$;
            sample the action with $\epsilon$-greedy from
              $Q_t^{\pi^i}(\hat{o}_t^i, u_t^i; \theta^i)$;
            Update RNN state $h_t^{(1)}, ..., h_t^{(n)} \leftarrow h_{t-1}^{(1)}, ..., h_{t-1}^{(n)}$;
            Receive reward $\boldsymbol{r}_t$ and observe next state $\mathbf{o}_{t+1}$;
            Add transition $\{\boldsymbol{o}_t, \boldsymbol{u}_t, \boldsymbol{r}_t, \boldsymbol{o}_{t+1}\}$ into $\mathcal{D}$;
        end
    end
    if $episodes > M$ then
        Sample a minibatch $\mathcal{B} = \{\boldsymbol{o}_j, \boldsymbol{u}_j, \boldsymbol{r}_j, \boldsymbol{o}_{j+1}\}_{j=0}^M \sim \mathcal{D}$;
        Update the policy $\pi^i$ using $\mathcal{B}$ and (5);
        Update the graph generator $\rho$ using $\mathcal{B}$ and $\nabla_\varphi L(\varphi)$;
        Update $\phi^i$ using (7);
        Every $C$ steps reset $\theta_i^- = \theta_i$ for $i = 1, ..., n$;
        Update the Lagrange penalty $\xi$ and the multipliers
          $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$;
    end
end
```

5.1 Experimental Setting

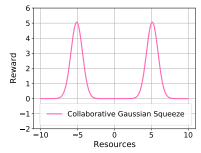
Collaborative Gaussian Squeeze (CGS). As an extension of Multi-domain Gaussian Squeeze (MGS) [9], Collaborative Gaussian Squeeze is a challenging environment for evaluating coordination. In MGS, there exist K domains $[(\mu_1, \sigma_1), ..., (\mu_K, \sigma_K)]$ in the system. The system contains N agents, and each agent i can take actions a_i within range of $\{-10, -9, .., 0, 1, .., 8, 9, 10\}$. The prior $s_i \in [0, 0.2]$ given by the environment represents the unit-level resource for each agent i. The total amount of resources mobilized by all agents is denoted as $f(\boldsymbol{u}) = \sum_i s_i \times a_i$. The goal is to maximize the joint reward $G(\boldsymbol{u}) = \sum_{k=1}^K f(\boldsymbol{u}) e^{-(f(\boldsymbol{u})-\mu_k)^2/\sigma_k{}^2}$. In our settings, we modify the original MGS to a collaborative task. We use two domains $[(\mu_1 = 5, \sigma_1 = 1.25), (\mu_2 = -5, \sigma_2 = 1.25)]$. The computation of the joint reward is as follows.

$$G(\boldsymbol{u}) = f(\boldsymbol{u})e^{-(f(\boldsymbol{u})-\mu_1)^2/\sigma_1{}^2} - f(\boldsymbol{u})e^{-(f(\boldsymbol{u})-\mu_2)^2/\sigma_2{}^2}. \tag{18}$$

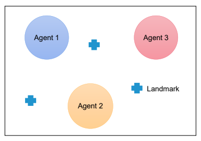
Figure 3a shows the reward curves for this setting. According to the above definition, the reward is maximized when the resources of all the agents reach μ_1 or μ_2. Social welfare depends on the intensity of collaboration.

Cooperative Navigation (CN). Cooperative Navigation is a classic scenario implemented in the multi-agent particle world. This scenario has n agents and n landmarks, which are initialized with random locations at the beginning of each episode. The objective of the agents is to cooperate to cover all landmarks by controlling their velocities with directions. The action set $\mathcal{A}$ includes 5 actions: `[up, down, left, right, stop]`. Each agent can only observe its velocity, position, and displacement from other agents and the landmarks. The shared reward is the negative sum of displacements between each landmark and its nearest agent. Agents must also avoid collisions, since each agent is penalized with a '-1' shared reward for every collision with other agents. We set the length of each episode as 25 time-steps. Therefore, the agents have to learn to navigate towards the landmarks cooperatively to cover all positions quickly and accurately. Figure 3b shows a classic scenario in Cooperative Navigation with $N = 3$.

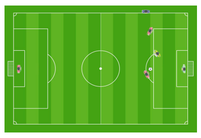
Google Research Football (GRF) . GRF is a realistically complicated and dynamic simulation environment without any clearly defined behavior abstraction, which is a suitable testbed for studying multi-agent decision-making and coordination. In GRF, we use the *Floats* wrappers to represent the state. The *Floats* representation contains a 115-dimensional vector that summarizes the information such as the ball position and possession, coordinates of all players, and the game state. Each player has 19 actions to control, including the standard move actions and different ball-kicking ways. The rewards include the *SCORING* reward $(-1, +1)$ and the *CHECKPOINT* reward, which is a shaped one that specifically addresses the sparsity of *SCORING*. Detailed descriptions are shown in Appendix B.



(a) CGS



(b) CN



(c) GRF

Figure 3: The schematics of our experimental environments.

Baselines. We compared our results with several baselines as follows. VDN and QMIX are the state-of-the-art value factorization approaches that follow the regime of centralized training and decentralized execution, belonging to fully decentralized control policies which are difficult to obtain coordinated behaviours. DCG uses fully connected graphs for belief propagation, which only allows the message passing of paired agents. DGN aims at learning abstract representations to make simultaneous decisions.

- VDN [6]: Value Decomposition Network (VDN) imposes the tructural constraints of the additivity in factorization, which represents Q_{tot} as a sum of individual Q-values.
- QMIX [7]: It is proposed to overcome the limitations that VDN has a strong assumption and ignores any extra state information available during training. QMIX enforces that Q_{tot} is monotonic in the individual Q-values Q^i.
- DCG [26]: Deep Coordination Graph (DCG) factorizes the joint value function of all agents according to a coordination graph into payoffs between pairs of agents, which coordinates the actions between agents explicitly.
- DGN [28]: DGN is to rely on graph convolutional network to model the relation representations, implicitly modeling the action coordination.

5.2 Main Results

Here report the experimental results on the settings described in Section 5.1, and performance validation indicates the superiority of introducing ACG to the multi-agent systems.

Collaborative Gaussian Squeeze. There are 10 agents cooperate to obtain the optimal team reward and the maximum episode length is set to 10. To emphasize the feasibility and effectiveness of our proposed framework, we firstly conduct the experiment on CGS. We report the average episode rewards over 10 random runs shown in Figure 4. Our proposed algorithm GCS outperforms baseline methods by a large margin. It is observed that our algorithm can handle the collaborative problem well, which benefits from the action coordination graph facilitating the behavior learning to promote cooperation. Next, we will verify the effectiveness of our algorithm on more complicated environments.

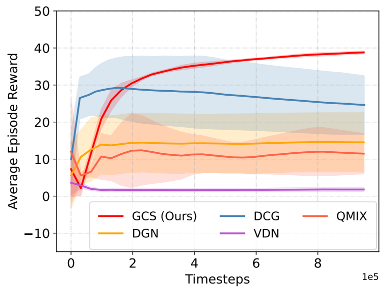
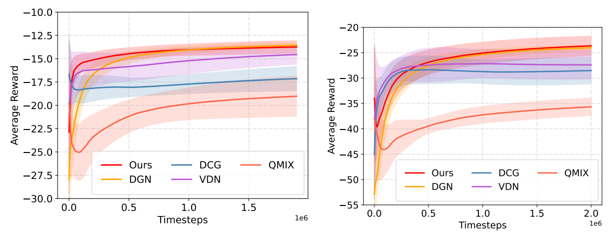


Figure 4: Average episode rewards comparing with the baselines on collaborative gaussian squeeze.

Cooperative Navigation. As shown in Figure 3b, Cooperative Navigation is a fully cooperative environment in which N agents (circles) must cooperate to reach N landmarks (crosses) with as few collisions as possible. We conduct experiments in the Cooperative Navigation with $N = 4$ and $N = 6$, respectively. Figures 5a and 5b show the learning curve comparisons for two cases. We report the average episode reward at a training step, averaged over 10 independent running seeds.

First, our algorithm outperforms the most baseline algorithms by higher converged rewards. We consider that our performance improvement results from the optimal use of the action coordination graph represent the action dependency for better coordination. Moreover, our algorithm converges fast during training, which is possible because the hierarchical decision policies can effectively induce coordination among agents in this cooperative setting. In addition, our algorithm can achieve a lower variance than those

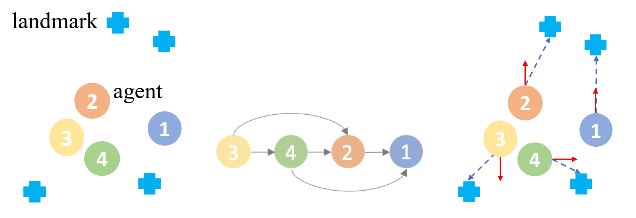


(a) 4 agents **(b) 6 agents**

Figure 5: Average episode rewards for 4 agents and 6 agents in Cooperative Navigation.

baselines, which indicates that the learned action coordination graph can reduce the uncertainty in decision-making to facilitate cooperative behaviors among agents.

In contrast, VDN and QMIX take actions simultaneously without considering the action dependency among agents. They are faster during training, but that is helpless in inducing cooperation among agents. Additionally, DCG exhibit mediocre performance in this task. We believe that DCG considers only the pairwise relationship between agents, which may disturb the overall balance in the system. In this case, DGN shows good performance consistent with ours, which shows that implicit action coordination modelling is also effective in pure cooperative settings.



(a) Current state **(b) Topology** **(c) Goals and actions**

Figure 6: Illustration of the effects of the action coordination graph. (a), Current state. (b), Topological structure of the learned ACG. (c), The intention for agents according to the ACG. The dashed lines represent the targets, and the solid lines represent the actions.

In order to clearly show the meaningful effect of the action coordination graph in the decision-making process, we give a visualization at a time step in an episode of Cooperative Navigation, as shown in Figure 6. Figure 6b shows the topological structure of the learned ACG, and it suggests the decision dependency of the agent is [3, 4, 2, 1]. The parent sets of the four agents are denoted as $pa(1) = \{4, 2\}$, $pa(2) = \{3, 4\}$, $pa(3) = \emptyset$, and $pa(4) = \{3\}$, respectively. First, agent 3 decides to move to the bottom-left landmark, then agent 4 takes the best response and decides to move to the bottom-right in order to avoid conflict with agent 3. After agent 2 knows the decisions of the previous two agents, it chooses the closer upper-left as its target instead of the bottom-right. Finally, agent 1 moves after observing the decisions of agents 2 and 4. This visualization shows how agents' joint actions deriving from ACG representing the underlying decision dependency achieves efficiency.

Google Research Football (GRF). For evaluating our method in complicated and dynamic environments, we conduct several experiments on GRF, as shown in Figure 7. In the 3-vs-2 scenario, three of our players try to score from the edge of the box, and the opponent team contains one defender and one keeper. In the 3-vs-6 scenario, there are six opponent players on the court to play against three of our players. In the 5-vs-5 scenario, each team has a keeper, an offensive player, and three defenders. Here, we report the average episode reward at a training step for each scenario, averaged over 10 independent running seeds.

As can be seen in Figure 7, our algorithm always obtains higher rewards than all the baselines in different scenarios of GRF. It indicates that our method is quite general against different scenarios in complicated and dynamic environment. Moreover, this performance improvement in GRF demonstrates our approach is good at effectively handling stochasticity and sparse rewards. This is because the learned ACG with the decision dependency is an efficient way to mitigate the uncertainty and induce cooperation among agents. Taking the 3-vs-6 scenario for further demonstration, the training curve of QMIX fluctuates and is unstable, indicating this method's incapability of adapting to the dynamically complicated scenario with multiple opponent players. Here, DCG shows a trend of non-convergence, but our algorithm steadily rises to converge and obtains the highest reward, which exhibits the modeling supremacy of ours for handling complicated tasks.

5.3 Results on DAG Depth

We observe that the inference efficiency and the performance gains are affected by the ACG's depth. Therefore, we propose to find the optimal depth k to balance them. As shown in Figure 8, to validate the impact of the depth, we test our method on the collaborative gaussian squeeze with different depth sizes of the learned ACG. In this figure, the horizontal axis is the depth, and the vertical axis is the testing episode reward averaged over 5 seeds. We test 1000 episodes for each seed and obtain an average episode reward.

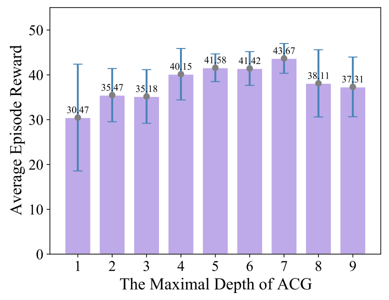
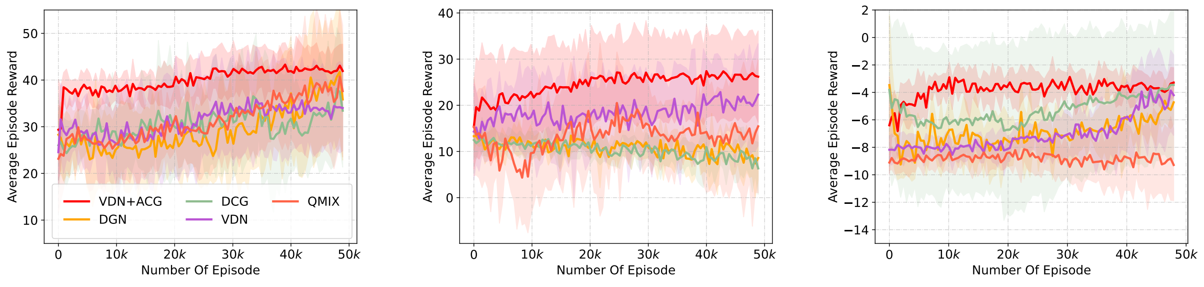


Figure 8: Constraining different DAG depth on collaborative gaussian squeeze.

As the depth of ACG increases, the training time will increase correspondingly. However, the performance growth will gradually slow down, and even performance degradation will occur. In this case, $k = 5$ is the optimal depth, which balances the computational burden and performance gains. As for the reason for the performance degradation with $k = 8$ and $k = 9$, we speculate that as



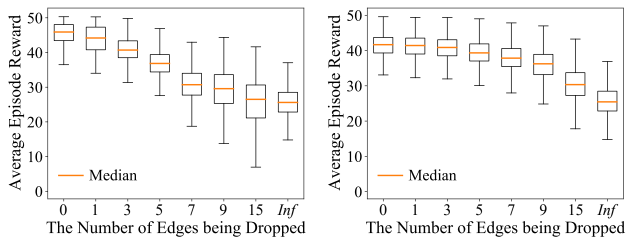
(a) 3 vs 2 (b) 3 vs 6 (c) 5 vs 5

Figure 7: Average episode reward vs. training steps for comparisons with the baselines on Google Research Football.

the hierarchy of action dependency deepens, the complexity of the hypothesis space for the inference will increase, and it is harder to learn the optimal policy. In summary, the higher dependency level of the graph structure can provide more decision information to promote coordination and facilitate performance. However, the higher dependency level leads to the lower efficiency inference, as the leaf node on the ACG needs to wait for all the parent nodes' decisions before it makes the decision.

5.4 Results on Dropping Edges

In this section, we verify the stability of the learned ACG in our proposed algorithm. Given a trained model with $depth = 5$ on collaborative gaussian squeeze, we evaluate 1000 episodes and count the average number of edges of ACG, denoted as $edges_{num} \approx 28$. A fair comparison requires the same depth and number of edges during training and evaluation. Therefore, we generate a fixed directed acyclic graph structure $\mathcal{G}_{\{5,28\}}$, see Appendix C, whose depth is 5 and whose number of edges is 28 as the baseline to compare with our algorithm.



(a) The baseline $\mathcal{G}_{\{5,28\}}$ **(b) GCS (Ours)**

Figure 9: The box plots showing the distribution of the testing episode rewards in Collaborative Gaussian Squeeze.

As shown in Figure 9, the horizontal axis represents the number of edges dropped from the generated or fixed graph structure, denoted as $\#drop$, and the vertical axis is the testing average episode reward over 1000 episodes. inf denote dropping all the edges. The box plot visually show the distribution of the testing episode rewards and skewness through displaying the data quartiles. From the overall trend of the Figure 9, we can observe that the data quartiles of the baseline reduces faster and changes more drastically than our algorithm when the disturbance of edge dropping increases. It demonstrates that our algorithm has better stability. Moreover, in Table 1, our algorithm outperforms the baseline with higher mean rewards in most cases, which benefits from the power of ACG learned by our model to promote the coordination among agents reliably and stably even when confronted with disturbances of different intensities. It is worth noting that for guaranteeing stability, a slight performance loss may occur. It will be an interesting direction to study the stability-performance trade-off in the future.

#drop / methods	0	1	3	5	7	9	15	inf
The baseline	45.3	44.0	40.7	36.9	30.8	29.3	25.8	26.2
GCS (Ours)	41.6	41.3	40.9	39.4	37.9	36.0	30.5	26.1

Table 1: The mean value of the testing episode rewards when dropping the corresponding number of edges.

In summary, comparisons with VDN, QMIX, and DCG on three environments demonstrate that our algorithm achieves better performance, stronger stability, and powerful modeling capability of handling dynamically complicated tasks. Moreover, the proposed DAG depth constraint provides an insightful view to balance efficiency and performance.

6 CONCLUSIONS

In this paper, we introduce the novel graph generator and factored coordination policy in MARL to dynamically represent the underlying decision dependency structure and facilitate behavior learning, respectively. We propose the DAGness and depth constrained optimization to balance the training efficiency and the performance gains. Extensive empirical experiments on Collaborative Gaussian Squeeze, Cooperative navigation, and Google Football as well as comparisons to the baseline algorithms demonstrate the superiority of our method.

Future research may consider improving the limited performance by upgrading the graph generator model. We will also investigate an automatic mechanism to find an appropriate depth for the action coordination graph.

ACKNOWLEDGEMENTS

This work was supported in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA27010404 and in part by the National Nature Science Foundation of China under Grant 62073324.

REFERENCES

[1] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[2] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. pages 6382–6393, 2017.

[3] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671, 2008.

[4] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 464–473, 2017.

[5] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970, 2019.

[6] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 2085–2087, 2018.

[7] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304, 2018.

[8] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Annual Conference on Neural Information Processing Systems*, 2020.

[9] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896, 2019.

[10] Yali Du, Lei Han, Meng Fang, Tianhong Dai, Ji Liu, and Dacheng Tao. Liir: learning individual intrinsic reward in multi-agent reinforcement learning. In *Annual Conference on Neural Information Processing Systems*, pages 4403–4414, 2019.

[11] Haifeng Zhang, Weizhe Chen, Zeren Huang, Minne Li, Yaodong Yang, Weinan Zhang, and Jun Wang. Bi-level actor-critic for multi-agent coordination. In *AAAI Conference on Artificial Intelligence*, pages 7325–7332, 2020.

[12] Dimitri Bertsekas. Multiagent rollout algorithms and reinforcement learning. *arXiv preprint arXiv:1910.00120*, 2019.

[13] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[14] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604–609, 2020.

[15] Duzhen Zhang, Tielin Zhang, Shuncheng Jia, Xiang Cheng, and Bo Xu. Population-coding and dynamic-neurons improved spiking actor network for reinforcement learning. *arXiv preprint arXiv:2106.07854*, 2021.

[16] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning*, pages 330–337, 1993.

[17] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, pages 2974–2982, 2018.

[18] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[19] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[20] Jakob N Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Annual Conference on Neural Information Processing Systems*, pages 2145–2153, 2016.

[21] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Annual Conference on Neural Information Processing Systems*, pages 2252–2260, 2016.

[22] Chongjie Zhang and Victor Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1101–1108, 2013.

[23] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *CoRR*, abs/1703.10069, 2017.

[24] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in Neural Information Processing Systems*, 31:7254–7264, 2018.

[25] Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. Learning correlated communication topology in multi-agent reinforcement learning. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 456–464, 2021.

[26] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pages 980–991, 2020.

[27] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 764–772, 2021.

[28] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*, 2018.

[29] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Annual Conference on Neural Information Processing Systems*, pages 9472–9483, 2018.

[30] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163, 2019.

[31] Yue Yu and Tian Gao. Dags with no curl: Efficient dag structure learning. In *Annual Conference on Neural Information Processing Systems*, 2020.

[32] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural dag learning. 2020.

[33] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. Causal discovery with reinforcement learning. In *International Conference on Learning Representations*, 2019.

[34] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, volume 96, pages 195–210, 1996.

[35] I. N. Herstein. *Topics In Algebra.* John Wiley & Sons, 2nd edition, 1975.

[36] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48:334–334, 1997.

[37] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods.* Academic press, 2014.

[38] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers.* Now Publishers Inc, 2011.

[39] A Nemirovski. *Optimization II numerical methods.* Citeseer, 1999.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[41] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

[42] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *AAAI Conference on Artificial Intelligence*, pages 4501–4510, 2020.

A DETAILED PROOFS

We provide the detailed proofs of the propositions in the following.

A.1 Proof of Proposition 1

PROOF. Firstly, we prove that the entry A_{ij}^{k} in k-th power of the adjacency matrix A indicates the number of walks of length k from node v_i to v_j. Let $\mathcal{L}^{l}(i,j)$ denote the number of walks of length l from node v_i to v_j. When $l = 1$, $A^1 = A, a_{ij} = \mathcal{L}^1(i,j)$. Let $b_{ij} = \mathcal{L}^k(i,j)$ denote the ij^{th} entry of A^k. We have $A_{ij}^{k+1} = A_{ij}A_{ij}^{k} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj} = \sum_n a_{in}b_{nj}$, where $\forall_m \in [1,n], a_{im}b_{mj} = \mathcal{L}^1(i,m) \cdot \mathcal{L}^k(m,j) = \mathcal{L}^{k+1}(i,j)$. So A_{ij}^{k+1} denote the number of walks of length $k+1$ from v_i to v_j. Then:

$$\begin{aligned} d(A^k) &:= sum(A^k) = \textstyle\sum_i \sum_j A_{ij}^{k} = 0 \\ d(A^{k-1}) &:= sum(A^{k-1}) = \textstyle\sum_i \sum_j A_{ij}^{k-1} > 0 \end{aligned} . \tag{19}$$

The equations represent there is not a walk of length k and there is at least one walk of length $k-1$ from v_i to v_j respectively, that is $A^k = O, A^{k-1} \neq O$. We define the hierarchy of the DAGs as the longest path length. Therefore, the hierarchy of the DAGs is k if A is the Nilpotent Matrix of index k. □

A.2 Proof of Proposition 2

PROOF. Let π^i represent the fixed policy for agent i trained by the factored coordination policy. φ is the parameter to be solved by the graph generator ρ. $g(\cdot)$ and $d(\cdot)$ are the constraint function as shown in Equation (8) and Equation (19). $Q_{\pi^i}(\cdot)$ denotes the state action function. The gradients is derived as follows.

$$\begin{aligned}
&\nabla_{\varphi} L(\varphi, \lambda_1, \lambda_2) = \nabla_{\varphi} \sum_{s} p_{\pi^i}(s) \sum_{W} \rho_{\varphi}(W|s) \sum_{u^i} \pi^i(u^i|s,W) \cdot Q_{\pi^i}(s,u^i) \\
&-\lambda_1 g(W) - \lambda_2 d(W^k)\Big] \\
&= \nabla_{\varphi} \sum_{s} p_{\pi^i}(s) \sum_{W} \rho_{\varphi}(W|s) \sum_{u^i} \Big[\pi^i(u^i|o^i, u^{pa(i)}) \cdot \big[Q_{\pi^i}(o^i,u^i) \\
&-\lambda_1 [tr(e^{W \circ W}) - d] - \lambda_2 sum(W^k)\Big] \\
&= \mathbb{E}_{s \sim p, W \sim \rho_{\varphi}(\cdot|s)} \Big[\nabla_{\varphi} \log \rho_{\varphi}(W|s) \sum_{u^i} \pi^i(u^i|o^i, u^{pa(i)}) \cdot Q_{\pi^i}(o^i,u^i) \\
&-\lambda_1 (e^{W \circ W})^T \cdot 2W - \lambda_2 \textstyle\sum_{i,j} \left[kW^k W^{-1}\right]_{ij}\Big]
\end{aligned}$$

□

B DEATILS OF GOOGLE RESEARCH FOOTBALL

Observations. The environment exposes the raw observations as Table 2. We use the *Simple115StateWrapper*[4] as the simplified representation of a game state encoded with 115 floats.

Information	Descriptions
Ball information	position of ball direction of ball rotation angles of ball owned team of ball owned player of ball
Left team	position of players in left team direction of players in left team tired factor of players yellow card numbers of players whether a player got a red card roles of players
Right team	position of players in right team direction of players in right team tired factor of players yellow card numbers of players whether a player got a red card roles of players
controlled player information	controlled player index designated player index active action
Match state	goals of left and right teams left steps current game mode
Screen	rendered screen

Table 2: The main information and the detailed descriptions about the observations in GRF.

Actions. The number of actions available to an individual agent can be denoted as $|\mathcal{A}| = 19$. The tandard move actions (in 8 directions) include $\mathcal{A}_{move} = \{Top, Bottom, Left, Right, Top{-}Left, Top{-}Right, Bottom{-}Left, Bottom{-}Right\}$. Moreover, the actions represent different ways to kick the ball is $\mathcal{A}_{kick} = \{ShortPass, HighPass,$ $LongPass, Shot, Do{-}Nothing, Sliding, Dribble, Stop{-}Dribble, Sprint, Stop-Moving, Stop-Sprint\}$.

[4]We refer the readers to:https://github.com/google-research/football for details encoded infromation.

Rewards. The reward function mainly includes two parts. The first is *SCORING*, which corresponds to the natural reward where the team obtains +1 when scoring a goal and −1 when losing one to the opposing team. The second part is *CHECKPOINT*, which is proposed to solve the sparse reward. It is encoded with the domain knowledge by the additional auxiliary reward contribution.

C THE DETAILED STRUCTURE OF $\mathcal{G}_{\{5,28\}}$

The adjacency matrix of $\mathcal{G}_{\{5,28\}}$ is shown as follows.

$$\begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0
\end{bmatrix}$$

D ADDITIONAL EXPERIMENTAL DETAILS

We set discount factor $\gamma = 0.99$. The optimization is conducted using RMSprop with a learning rate of 5×10^{-4} and $\alpha = 0.99$ with no weight decay. Exploration for action selection is performed during training, and each agent executes $\epsilon - greedy$ policy over its actions. ϵ is annealed from 0.2 to 0.05 over the first $50k$ time steps and is kept constant afterwards.

In addition, the information regarding computational resources used is Enterprise Linux Server with 96 CPU cores and 6 Tesla K80 GPU cores(12G memory).