Sparse Action-Dependent Policy Iteration under Coordination Structures: Convergence and Optimality

Jianglin Ding[1] **Jingcheng Tang**[1] **Gangshan Jing**[*1]

[1]School of Automation, Chongqing University, Chongqing, China

Abstract

Action-dependent policies, which condition decisions of each agent on both states and other agents' actions, provide a powerful structured framework for cooperative multi-agent reinforcement learning (MARL). Most existing studies have focused on auto-regressive formulations, where each agent's policy depends on the actions of all preceding agents. However, this structure suffers from severe scalability limitations as the number of agents grows. In contrast, the theoretical foundations of sparse dependency structures remain largely unexplored. To address this gap, we introduce the Action Dependency Graph (ADG) to model sparse inter-agent dependencies. We propose a refined equilibrium concept with respect to the ADG that is stronger than the Nash equilibrium which often traps independent policies. Furthermore, within Coordination Graphs (CG) structured problems, we show that such an equilibrium attains global optimality when the ADG satisfies specific CG-induced conditions. To substantiate our theory, we develop a tabular multi-agent policy iteration algorithm that converges to the refined equilibrium exactly as predicted. We further extend our approach to deep MARL, confirming that these structural conditions provide a reliable design principle for scalable and optimal coordination.

1 INTRODUCTION

Achieving effective multi-agent reinforcement learning (MARL) in fully cooperative environments requires agents to coordinate their actions to maximize collective performance. While independent policies [Zhang et al., 2021, Oroojlooy and Hajinezhad, 2023] offer computational scalability, they implicitly assume a fully factorized joint policy, ignoring the complex dependencies required for global coordination. Although computationally tractable, these completely decentralized representations are often suboptimal [Fu et al., 2022], as they tend to converge to one of many Nash equilibrium solutions [Ye et al., 2022] that may not correspond to the globally optimal configuration.

The emergence of action-dependent policies [Fu et al., 2022] offers a promising solution by reframing coordination as a structured decision process. In this context, we introduce the action dependency graph (ADG), a directed acyclic graph that defines the conditional independence among agents' actions. Theoretical studies [Bertsekas, 2021, Chen and Zhang, 2023] demonstrate that policies in the auto-regressive form, associated with fully dense ADGs, can recover the full joint distribution and guarantee global optimality. However, fully dense ADGs pose substantial scalability issues, as the dependency complexity renders them intractable for large-scale systems. The fundamental challenge lies in identifying sparse ADG structures that preserve optimality guarantees while maintaining efficient inference.

This leads to two fundamental challenges that remain unaddressed in the literature: establishing convergence guarantees for policies defined by arbitrary sparse ADGs, and identifying the conditions under which such sparse ADGs can maintain global optimality guarantees. While the convergence properties of independent and fully auto-regressive policies are well-documented, the behavior of policies with general dependency structures represents a significant theoretical gap.

We address these challenges by unifying ADGs with the framework of Coordinated Reinforcement Learning [Guestrin et al., 2002], where a coordination graph (CG) represents the additive decomposition of the joint utility function. We establish a convergence result for policy iteration, proving that iterating any action-dependent policy converges to a G_d-locally optimal policy, a structural equilibrium where each agent's conditional decision maximizes

[*]Correspondence to Gangshan Jing: jinggangshan@cqu.edu.cn

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:1423–1447.

the global objective given the fixed actions of its ADG parents. Our key insight is that the permissible sparsity of an ADG, which ensures that such a G_d-locally optimal policy is guaranteed to be globally optimal, is governed by its structural compatibility with the CG. Building on this foundation, we derive the structural conditions relative to the CGs under which this convergence further extends to global optimality.

The contributions of this paper are summarized as follows.

(i) We introduce the notion of G_d-local optimality, a structural refinement of the Nash equilibrium that precisely characterizes the convergence behavior of action-dependent policies associated with the ADG G_d.

(ii) We establish a theoretical framework that unifies CGs with ADGs, deriving the first set of structural conditions under which sparse ADGs preserve global optimality.

(iii) We design a policy iteration algorithm that, grounded in our theory, guarantees convergence to a G_d-locally optimal policy, and further to a globally optimal solution when the proposed conditions on the CG and ADG are satisfied.

2 RELATED WORK

Independent policy. The majority of the literature on MARL represents the joint policy as the Cartesian product of independent individual policies. Value-based methods such as IQL [Tan, 1993], VDN [Sunehag et al., 2018], QMIX [Rashid et al., 2020], and QTRAN [Son et al., 2019] employ local value functions that depend only on the state or observation of each agent. Similarly, policy-based methods such as MADDPG [Lowe et al., 2017], COMA [Foerster et al., 2018], MAAC [Iqbal and Sha, 2019], and MAPPO [Yu et al., 2022] directly adopt independent policies. These approaches often fail to achieve global optimality, as they are not able to cover all strategy modes [Fu et al., 2022].

Coordination graph. Some value-based methods [Böhmer et al., 2020, Castellini et al., 2021, Li et al., 2021, Wang et al., 2022b] recognize that the limitation of independent policies is due to a game-theoretic pathology known as relative overgeneralization [Panait et al., 2006]. To mitigate this, they employ a higher-order value decomposition framework by introducing the coordination graph (CG) [Guestrin et al., 2002]. In this graph, the vertices represent agents, and the edges correspond to pairwise interactions between agents in the local value functions. While CGs improve cooperation by considering inter-agent dependencies, the resulting joint policy cannot be decomposed into individual policies. Consequently, decision-making algorithms still require intensive computation, such as Max-Plus [Rogers et al., 2011] or Variable Elimination (VE) [Bertele and Brioschi, 1972]. When the CG is dense, these computations may become prohibitively time consuming, making the policy difficult to execute in real time.

Action-dependent policy. In contrast to independent policies, action-dependent policies [Wang et al., 2022a, Ruan et al., 2022, Li et al., 2023, 2024] incorporate not only the state, but also the actions of other agents into an agent's decision-making process. The action dependencies among agents can be represented by a directed acyclic graph, which we refer to as the action dependency graph (ADG). In some literature, the action-dependent policy is also referred to as Bayesian policy [Chen and Zhang, 2023] or auto-regressive policy [Fu et al., 2022]. Moreover, the use of action-dependent policies can be viewed as a mechanism to leverage communications for enhancing cooperation [Zhou et al., 2023, Duan et al., 2024, Jing et al., 2024, Tang et al., 2025]. Some approaches [Bertsekas, 2021, Ye et al., 2022, Wen et al., 2022] transform a multi-agent MDP into a single-agent MDP with a sequential structure, enabling each agent to consider the actions of all previously decided agents during decision-making. This transformation ensures the convergent joint policy to be globally optimal [Bertsekas, 2021]. However, the fully dense ADG makes these methods computationally expensive and impractical for large-scale systems. For more general ADGs, existing theories can only guarantee convergence to a Nash equilibrium solution [Chen and Zhang, 2023]. Currently, no theoretical evidence demonstrates the superiority of action-dependent policies with sparse dependency graphs over independent policies.

3 PRELIMINARY

We formulate the cooperative multi-agent reinforcement learning problem as a *Multi-Agent Markov Decision Process* (MAMDP), represented by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{N} = \{1, \dots, n\}$ denotes the set of agents, $\mathcal{S}$ is the finite state space, $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}_i$ is the joint action space formed by the Cartesian product of each agent's finite action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor.

We consider policies of the deterministic form $\pi : \mathcal{S} \to \mathcal{A}$. The state value function and state-action value function induced by a policy π are

$$V^{\pi}(s) := \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t) \,\middle|\, s^0 = s\right], \tag{1}$$

$$Q^{\pi}(s, a) := \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t) \,\middle|\, s^0 = s,\, a^0 = a\right], \tag{2}$$

where the expectation $\mathbb{E}$ is taken over all random variables s^t induced by π and P.

For any function $V \in \mathcal{R}(\mathcal{S})$, where $\mathcal{R}(\mathcal{S})$ denotes the set

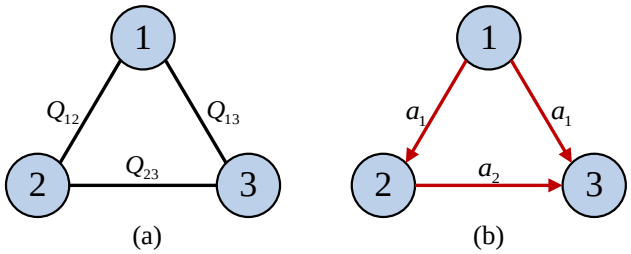

Figure 1: A coordination graph (a) and an action dependency graph (b).

of real-valued functions $J : \mathcal{S} \to \mathbb{R}$, we define

$$Q^V(s,a) := r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a)V(s'). \quad (3)$$

The Bellman operator $\mathcal{T}_\pi : \mathcal{R}(\mathcal{S}) \to \mathcal{R}(\mathcal{S})$ and the Bellman optimality operator $\mathcal{T} : \mathcal{R}(\mathcal{S}) \to \mathcal{R}(\mathcal{S})$ are given by

$$\mathcal{T}_\pi V(s) = Q^V(s,\pi(s)), \quad \mathcal{T}V(s) = \max_{a \in \mathcal{A}} Q^V(s,a), \quad (4)$$

respectively. The value function V^π is the unique fixed point of $\mathcal{T}_\pi$, and the optimal value function V^* is the unique fixed point of $\mathcal{T}$.

3.1 COORDINATION GRAPH

In many practical scenarios such as sensor networks [Zhang and Lesser, 2011], wind farms [Bargiacchi et al., 2018], mobile networks [Bouton et al., 2021], etc., the Q-function can be approximated as the sum of local value functions, each depending on the states and actions of a subset of agents. A widely used approach to representing this decomposition is the use of the *coordination graph* (CG) [Guestrin et al., 2002], which captures the pairwise coordination relationships between agents. Formally, we define a CG as follows.

Definition 3.1 (Coordination Graph)**.** An undirected graph[1] $G_c = (\mathcal{N}, \mathcal{E}_c)$ is a CG of a value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, if there exists a local function $Q^\pi_{ij} : \mathcal{S} \times \mathcal{A}_i \times \mathcal{A}_j \to \mathbb{R}$ for every edge $(i,j) \in \mathcal{E}_c$, and a local function $Q^\pi_i : \mathcal{S} \times \mathcal{A}_i \to \mathbb{R}$ for every vertex $i \in \mathcal{N}$, such that for any $a \in \mathcal{A}$, the following decomposition holds:

$$Q^\pi(s,a) = \sum_{i \in \mathcal{N}} Q^\pi_i(s,a_i) + \sum_{(i,j) \in \mathcal{E}_c} Q^\pi_{ij}(s,a_i,a_j). \quad (5)$$

Remark 3.2. If G_c is a subgraph of G'_c, and G_c is a CG of Q^π, then G'_c is also a CG of Q^π. Therefore, multiple CGs may correspond to the same value function Q^π.

Figure 1 (a) shows a CG where Q^π can be decomposed as:

$$Q^\pi(s,a) = Q^\pi_{12}(s,a_1,a_2) + Q^\pi_{13}(s,a_1,a_3) + Q^\pi_{23}(s,a_2,a_3). \quad (6)$$

[1]In this paper, the vertices and edges of the graph are represented by agent indices and index pairs.

3.2 NOTATIONS

In the paper, we frequently use sets as subscripts in expressions. Let $S \subseteq \mathcal{N}$, and denote its elements in ascending order as $S = \{s_1, s_2, \ldots, s_k\}$. For a space, such as $\mathcal{A}_S$, we define $\mathcal{A}_S := \prod_{i \in S} \mathcal{A}_i := \prod_{i=1}^k \mathcal{A}_{s_i}$. For a vector, such as a_S, we define $a_S := (a_{s_1}, a_{s_2}, \ldots, a_{s_k})$, $a_{s_i} \in \mathcal{A}_{s_i}$. The notation $< i$ indicates the set of agents with indices smaller than i, similarly for $\leq i$, $> i$, and $\geq i$. For an undirected graph G_c, $N_{G_c}(i)$ denotes the neighbor of vertex i. When there is no ambiguity, we abbreviate $N_{G_c}(i)$ as $N_c(i)$. $N_c[i] := N_c(i) \cup i$. For a directed graph G_d, $N_{G_d}(i)$ denotes the parent set of vertex i. Likewise, we abbreviate $N_{G_d}(i)$ as $N_d(i)$ and $N_d[i] := N_d(i) \cup i$.

4 ADG WITH OPTIMALITY GUARANTEE

4.1 ACTION DEPENDENCY GRAPH

In MARL, a deterministic joint policy $\pi(s)$ is a vector-valued mapping from states to actions, where each component corresponds to an individual action. Thus, $\pi(s)$ can always be written as a collection of independent policies $(\pi_1(s), \pi_2(s), \ldots, \pi_n(s))$. In this sense, the independent policies are expressive enough to form any joint deterministic policy, including the optimal one. However, this does not imply that independent learning can converge to the optimal joint policy, since independent policies cannot capture coordinated behaviors that rely on correlated actions across agents. The absence of such coordination may cause independent learners to converge to suboptimal points.

To address this limitation, we introduce a broader class of policies, termed *action-dependent policies*, whose inputs include not only the state but also the actions of other agents. Formally, specifying action-dependent policies requires determining the order in which actions are generated. Without loss of generality, we assume that actions are output sequentially according to agent indices $1, 2, \ldots, n$. In this case, the general form of agent i's policy is $\pi_i : \mathcal{S} \times \mathcal{A}_{<i} \to \mathcal{A}_i$. Since an agent's policy may not strictly require the actions of all preceding agents, we can represent these sparse action dependencies using a directed acyclic graph (DAG). Given the fixed generation order, we naturally assume throughout this paper that the topological order of any such DAG aligns with the agent indices, meaning that all directed edges (j,i) satisfy $j < i$.

Definition 4.1 (Action Dependency Graph (ADG))**.** A DAG $G_d = (\mathcal{N}, \mathcal{E}_d)$ is the Action Dependency Graph of a joint action-dependent policy $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ if, for every agent $i \in \mathcal{N}$, the policy π_i depends only on the state s and the actions of its parent set $N_d(i)$ in G_d. Formally, for any $s \in \mathcal{S}$ and $a_{<i}, a'_{<i} \in \mathcal{A}_{<i}$, if $a_{N_d(i)} = a'_{N_d(i)}$, then

$\pi_i(s, a_{<i}) = \pi_i(s, a'_{<i})$.

For convenience, we sometimes directly denote the individual policy as $\pi_i(s, a_{N_d(i)})$ to explicitly reflect this sparse dependency. Figure 1(b) illustrates the ADG of a joint policy π with the following form:

$$\pi(s) = (\pi_1(s), \pi_2(s, \pi_1(s)), \pi_3(s, \pi_2(s, \pi_1(s)), \pi_1(s))) . \tag{7}$$

From (7), it is evident that expressing the components of a joint policy directly in terms of action-dependent policies becomes cumbersome. To streamline such representations, given $C \subseteq \mathcal{N}$, we recursively define the following policy notation:

$$\pi_{i,C}(s, a_C) = \begin{cases} a_i & \text{if } i \in C, \\ \pi_i(s, \pi_{<i,C}(s, a_C)) & \text{otherwise,} \end{cases} \tag{8}$$

where $\pi_{<i,C} = (\pi_{1,C}, \ldots, \pi_{i-1,C})$. Conceptually, $\pi_{i,C}$ is derived from π_i by fixing the actions of agents in C as given inputs, while the actions of the remaining preceding agents in $(< i) \setminus C$ are sequentially generated by their respective policies. Consequently, $\pi_{i,C}$ depends strictly on the actions in C. A special case is $\pi_{i,\varnothing}$, where no other agent's action is fixed as a given input. In this case, $\pi_{i,\varnothing}$ is exactly the individual component of π, and (7) can be rewritten concisely as

$$\pi(s) = (\pi_{1,\varnothing}(s), \pi_{2,\varnothing}(s), \pi_{3,\varnothing}(s)) . \tag{9}$$

4.2 COORDINATION POLYMATRIX GAME

A key reason why independent policies often converge to locally optimal solutions is the existence of Nash equilibrium policies [Zhang et al., 2022, Kuba et al., 2022], also known as agent-by-agent optimal policies [Bertsekas, 2021]. In this subsection, we illustrate the suboptimality of Nash equilibria through an example of *coordination polymatrix game* [Cai and Daskalakis, 2011], and demonstrate how action-dependent policies can overcome this limitation.

A coordination polymatrix game can be viewed as a single-step decision problem, formulated by a MAMDP tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, with $\mathcal{S} = \varnothing$ and $\gamma = 0$. In addition, the game is equipped with an undirected graph $G_c = (\mathcal{N}, \mathcal{E}_c)$ and a set of pairwise payoff functions $\{r_{ij}\}_{(i,j)\in\mathcal{E}_c}$, which together determine the global reward $r(a) = \sum_{(i,j)\in\mathcal{E}_c} r_{ij}(a_i, a_j)$. In this setting, r is equivalent to the (state)-action value function $Q : \mathcal{A} \to \mathbb{R}$, and G_c serves as the CG of Q. Figure 2 illustrates a polymatrix game with three agents, each having two possible actions, $\mathcal{A}_i = \{0, 1\}, i = 1, 2, 3$. The payoff matrices specify the reward for each agent pair; for example, if agents 1 and 2 both choose action 0, they receive a payoff of 1 together.

For independent policies, the joint policies $\pi = (1, 1, 1)$ and $\pi = (0, 0, 0)$ are both Nash equilibria. However, only

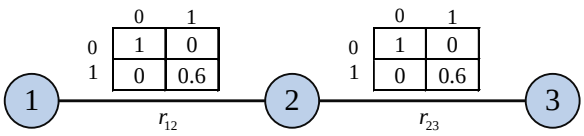

Figure 2: A polymatrix game on a line CG.

$(0, 0, 0)$ is globally optimal. Although $(1, 1, 1)$ is suboptimal, no single agent has an incentive to deviate unilaterally, since any individual deviation reduces the total reward.

Now consider action-dependent policies with an ADG G_d whose edge set is $\mathcal{E}_d = \{(1, 2), (2, 3)\}$. Suppose the policies are given by $\pi_1 = 1$, $\pi_2(0) = 0$, $\pi_2(1) = 1$, $\pi_3(0) = 0$, and $\pi_3(1) = 1$. This corresponds to the same joint policy $\pi = (1, 1, 1)$. However, if agent 1 switches its action to 0, agents 2 and 3 will also switch to 0, leading to the joint action $(0, 0, 0)$ with reward $r(0, 0, 0) = 2$, which exceeds $r(1, 1, 1) = 1.2$. Thus, agent 1 is incentivized to choose action 0, driving the system toward the globally optimal policy. This example shows exactly the advantage of using action-dependent policies.

4.3 OPTIMALITY GUARANTEE

Although action-dependent policies can converge to solutions stronger than Nash equilibria, such solutions are not necessarily globally optimal. In this subection, we introduce *G_d-local optimality* to characterize the optimality of action-dependent policies.

Definition 4.2 (G_d-locally Optimal)**.** Let $G_d = (\mathcal{N}, \mathcal{E}_d)$ be an ADG. A joint action-dependent policy $\pi = (\pi_1, \ldots, \pi_n)$ is G_d-locally optimal if, for every state $s \in \mathcal{S}$ and any $a_{N_d(i)} \in \mathcal{A}_{N_d(i)}$, the following holds:

$$\begin{aligned} &Q^\pi(s, \pi_{\mathcal{N}, N_d(i)}(s, a_{N_d(i)})) \\ &= \max_{a_i \in \mathcal{A}_i} Q^\pi(s, \pi_{\mathcal{N}, N_d[i]}(s, a_{N_d[i]})), \end{aligned} \tag{10}$$

where $\pi_{\mathcal{N}, N_d(i)}(s, a_{N_d(i)})$ represents the joint action that the actions of agents in the parent set $N_d(i)$ are fixed to $a_{N_d(i)}$, and the actions of other agents are sequentially generated by the remaining policy components. Because our recursive definition explicitly dictates that $\pi_{j, N_d(i)}(s, a_{N_d(i)}) = a_j, \forall j \in N_d(i)$, this notation elegantly captures the composite joint action $(a_{N_d(i)}, \pi_{-N_d(i), N_d(i)}(s, a_{N_d(i)}))$. When G_d is empty (i.e., independent policies), the G_d-local optimality coincides with agent-by-agent optimality. As more edges are added to G_d, condition (10) becomes increasingly restrictive. In the extreme case where G_d is a fully dense DAG with edge set $\mathcal{E}_d = \{(i, j) \in \mathcal{N} \times \mathcal{N} : i < j\}$, a G_d-locally optimal policy aligns with the globally optimal policy. A joint policy under policy iteration with ADG G_d

tends to converge to a G_d-locally optimal solution, as discussed in the next section. Thus, the most straightforward way to avoid suboptimality is to adopt a fully dense ADG. However, the computational cost of training and executing such policies grows rapidly with the number of agents, limiting scalability.

To identify sparse ADGs that preserve global optimality without prohibitive computational costs, we examine the structural dependencies required for maximizing the joint utility. We introduce the notion of a compatible ADG based on the sequential decomposition of the objective function.

Definition 4.3 (Compatible ADG)**.** Let G_b be a DAG over $\mathcal{N}$ without immoralities (i.e., for any node i, its parent set $N_b(i)$ forms a fully dense subgraph in G_b). A real-valued function $f : \mathcal{A} \to \mathbb{R}$ over the joint action space is said to additively *decompose according to* G_b if there exist local functions f_i such that

$$f(a) = \sum_{i \in \mathcal{N}} f_i(a_{N_b(i)}, a_i). \tag{11}$$

An ADG G_d is *compatible* with f if there exists such a DAG G_b over which f decomposes, and $G_b \subseteq G_d$ (i.e., $N_d(i) \supseteq N_b(i)$ for all $i \in \mathcal{N}$).

This definition bridges the agent's action dependencies with the underlying utility structure. Based on this, we present our main optimality theorem.

Theorem 4.4 (Optimality of Compatible ADG, proof in Appendix A)**.** *Let G_d be an ADG and the policy π be G_d-locally optimal. If G_d is compatible with the function $Q^\pi(s, \cdot)$ for all states $s \in \mathcal{S}$, then π is globally optimal.*

Proof Sketch. The proof proceeds by backward induction from agent n down to 1. Because the underlying DAG G_b has no immoralities, maximizing the objective over agent i's action perfectly absorbs the resulting dependencies into its fully connected parents $N_b(i)$, without creating any new fill-edges. Since the ADG is compatible ($N_d(i) \supseteq N_b(i)$), the required parent actions are already observed by agent i. Consequently, the G_d-local optimality condition ensures that each agent's conditional decision effectively executes a lossless variable elimination step, guaranteeing global maximum. □

Remark 4.5. For notational clarity, we formulate G_d and G_c as fixed global structures. However, the extension to state-dependent graphs, $G_d(s)$ and $G_c(s)$, is mathematically trivial. Because the joint maximization over actions is performed point-wise for each state s independently, there is no cross-state coupling. As long as $G_d(s)$ is compatible with $Q^\pi(s, \cdot)$ for each respective s, the sequential maximization guarantees hold independently at every state. To simplify the presentation, we consistently assume fixed global structures throughout the remainder of this paper.

Note that any arbitrary Q-function admits a trivial compatible DAG. For instance, a fully dense DAG is universally compatible by assigning the entire objective to the final agent (i.e., $f_n(a_{N_b(n)}, a_n) = Q^\pi(s, a)$ and $f_i = 0$ for $i < n$). This recovers the known result that fully dense ADGs guarantee global optimality. The non-trivial challenge, however, lies in identifying a *sparse* latent DAG G_b that preserves this guarantee. Fortunately, when the problem is structured by a CG, we can construct such a sparse compatible ADG by leveraging the structural equivalence between Markov networks and Bayesian networks [Theorem 4.13, Koller and Friedman, 2009], as formalized in the following corollary.

Corollary 4.6 (Optimality via CGs, proof in Appendix A)**.** *Let G_c be the CG of Q^π, and let G_h be any chordal completion of G_c. There exists a DAG G_b without immoralities that shares the exact same set of edges as G_h. Consequently, for any policy π with ADG G_d satisfying $G_b \subseteq G_d$, if π is G_d-locally optimal, then π is globally optimal.*

This corollary provides a practical blueprint for designing optimal ADGs directly from underlying CGs. Two special cases illustrate this principle: (i) When G_c is empty, it is already chordal. The corresponding G_b is empty, allowing G_d to be empty. This yields a VDN-style decomposition where any Nash equilibrium is globally optimal, consistent with Dou et al. [2022]. (ii) When G_c is a complete graph, its only chordal completion is itself. The resulting G_b is a fully dense DAG. Thus, for any arbitrary CG, a fully dense ADG guarantees global optimality, since every CG is a subgraph of the complete graph.

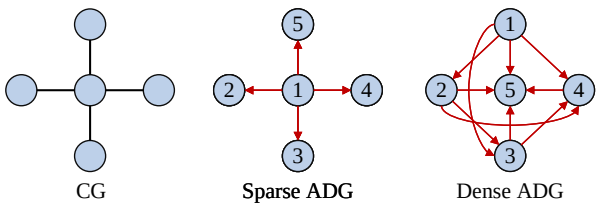

Figure 3: For a star CG, both the complete graph and itself are valid chordal completions. However, the ADGs they induce exhibit significantly different levels of sparsity.

For general CGs, different chordal completions can significantly impact the sparsity of the resulting ADG, as illustrated in Figure 3. The choice of the chordal completion G_h fundamentally governs the structure of the resulting G_b. If we measure sparsity in the sense of the parent set size, the sparsest compatible DAG G_b is obtained by finding a chordal completion G_h that minimizes the size of its maximum clique. Crucially, the minimum over all chordal completions of the size of the maximum clique (minus one) is precisely the definition of the treewidth of G_c [Koller and Friedman, 2009]. Thus, the maximum parent set size $|N_b(i)|$

in the sparsest possible ADG is bounded by the treewidth of the CG. For example, if G_c is a tree, its treewidth is 1, meaning each agent in G_b (and thus in G_d) depends on at most one other predecessor. However, finding the optimal chordal completion that minimizes this maximum clique size is an NP-complete problem [Kok and Vlassis, 2006], equivalent to finding the optimal elimination order in variable elimination (VE). Despite this complexity, practical heuristics from probabilistic inference, such as the min-fill elimination [Koller and Friedman, 2009], can be employed. Our greedy method (Algorithm 2 in Appendix C) instantiates this heuristic, efficiently generating sparse ADGs. Figure 4 illustrates the resulting ADGs for several simple CG topologies, which for these cases also happen to be the sparsest possible ADGs.

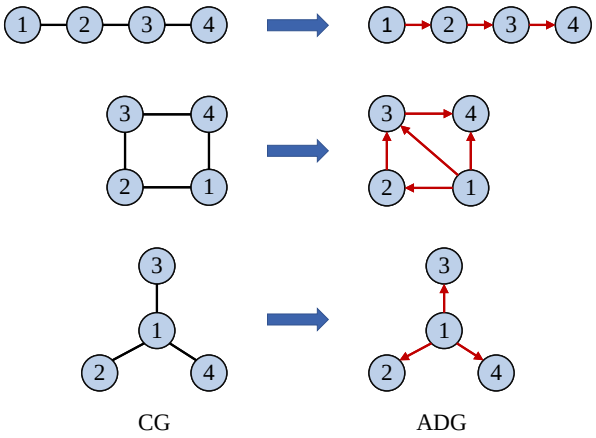

Figure 4: ADGs generated by Algorithm 2 for CG topologies: line, ring, and star.

5 CONVERGENCE OF ACTION-DEPENDENT POLICY

5.1 CONVERGENCE TO G_d-LOCALLY OPTIMAL POLICY

In this section, we introduce a policy iteration algorithm in the tabular setting. This algorithm highlights the advantage of employing action-dependent policies, enabling convergence to a G_d-locally optimal policy rather than merely an agent-by-agent optimal one.

Our approach Algorithm 1 extends the multi-agent policy iteration (MPI) framework proposed in Bertsekas [2021], which decomposes the joint policy update step of standard policy iteration (PI) [Sutton, 2018] into sequential updates of individual agents' policies, thereby mitigating the computational complexity of PI. However, MPI guarantees convergence only to an agent-by-agent optimal policy, which is often suboptimal. To address this limitation, we incorporate action-dependent policies into the MPI framework and ensures convergence to a G_d-locally optimal policy.

Algorithm 1 Action-Dependent Multi-Agent Policy Iteration

Initialize policies $\pi_i^1,\ i \in \mathcal{N}$, with ADG G_d under every $s \in \mathcal{S}$
for $k = 1, 2, \ldots$ **until** $\pi_i^k = \pi_i^{k-1}, \forall i \in \mathcal{N}$ **do**
  // Policy Evaluation
  Compute V^{π^k} by solving $V = \mathcal{T}_{\pi^k} V$
  // Policy Improvement
  for $i = 1, 2, \ldots, n$ **do**
    For every $(s, a_{N_d(i)})$ pair, compute the Q-value:

$$\hat{Q}_i(s, a_{N_d(i)}) \leftarrow Q^{\pi^k}(s, \pi^{k+1}_{(<i),N_d(i)}(s, a_{N_d(i)}), a_i, \pi^k_{(>i),N_d[i]}(s, a_{N_d[i]})) \tag{12}$$

    if $\pi_i^k(s, a_{<i}) \in \arg\max_{a_i} \hat{Q}_i(s, a_{N_d(i)})$ **then**
      $\pi_i^{k+1}(s, a_{<i}) \leftarrow \pi_i^k(s, a_{<i})$
    else
      $\pi_i^{k+1}(s, a_{<i}) \leftarrow \arg\max_{a_i} \hat{Q}_i(s, a_{N_d(i)})$
    end if
  end for
end for

Since the policies of agents in $(< i)$ are always updated before agent i, the (12) can always be applied in succession. Since the image set of $\arg\max$ may have multiple values, we arbitrarily select one of them. Specifically, if $\pi_i^k(s, a_{<i})$ is already in the image set, then we prioritize selecting $\pi_i^k(s, a_{<i})$. Note that while the update is specified for every $(s, a_{<i})$ pair, the $\arg\max$ in Algorithm 1 only depends on $(s, a_{N_d(i)})$, so the actual computation only needs to be performed for every $(s, a_{N_d(i)})$ pair. When update rule no longer changes π_i^k, the algorithm reaches convergence. The following theorem describes the convergence property:

Theorem 5.1 (Convergence of Algorithm 1, proof in Appendix B)**.** *Let $\{\pi^k\}_{k=1}^{\infty}$ be the policy sequence generated by Algorithm 1. Then $\{\pi^k\}_{k=1}^{\infty}$ converges to a G_d-locally optimal policy in a finite number of steps.*

Proof Sketch. The proof is structured to propagate stability from independent base cases to complex conditional dependencies through the following steps:

Step 1: Monotonic Convergence of Independent Components. We establish the monotonic convergence of the value function sequence $\{V^{\pi^k}\}$ (Lemma B.2). Since the policy space is finite, the non-decreasing nature of the value function ensures that the base independent components of the policy, $\pi_{\mathcal{N},\varnothing}$, must stabilize within finite iterations.

Step 2: Constructing a Total Order on the Power Set. To handle the potential oscillations of conditional policies

$\pi_i(s, a_{N_d(i)})$, we define a strict total order $<$ on the power set of predecessors $\mathcal{C} = \mathcal{P}(\{1, \ldots, n-1\})$ (Definition B.3). This ordering allows us to perform a structural induction on the complexity of the dependency context rather than just the number of agents.

Step 3: Handling Sparsity via Dependency Reduction. The core technical difficulty of the convergence of $\pi_{i,C}, C \in \mathcal{C}$ lies in sparse ADGs where an agent i may not observe certain predecessors in a given context C_m ($N_d(i) \not\supseteq C_m$). Lemma B.6 provides a reduction principle: if an agent's ADG does not cover the current context C_m, its policy behavior can be mapped back to a observable context C_j that precedes C_m in our total order. This ensures that the induction remains well-defined even when ADG are not fully dense.

Step 4: Completion of Induction. By inducting over the ordered contexts $C_m \in \mathcal{C}$, we prove that a change in any conditional policy would imply a strict increase in the already-converged value function, leading to a contradiction (Lemma B.7). Thus, the entire individual policies sequence must converge to a G_d-locally optimal limit. □

It is straightforward to verify that once all individual policies converge, the joint policy is G_d-locally optimal. However, the convergence of individual policies under general ADG structures presents significant theoretical challenges. Most existing literature focuses on the two extremes: independent policies or fully dense auto-regressive formulations. While Chen and Zhang [2023], Chen et al. [2024] demonstrated that action-dependent policies can converge to a Nash equilibrium under policy gradient methods, such results only ensures the stability of the marginalized independent components $\pi_{\mathcal{N},\varnothing}(s)$. The full conditional policies $\pi_i(s, a_{N_d(i)})$ may still exhibit persistent oscillations, preventing the joint police from stabilizing at a G_d-locally optimal point.

5.2 CONVERGENCE TO GLOBALLY OPTIMAL POLICY

When the CG of an MDP is fixed (e.g., polymatrix games), we can construct an ADG G_d that is compatible with the value function Q^π structured by a CG G_c (via the chordal completion outlined in Corollary 4.6), and we simply say that G_d is compatible with the CG G_c. Then, apply Algorithm 1 to update policies under this ADG. Upon convergence, Theorem 5.1 ensures that the resulting policy is G_d-locally optimal, and by Corollary 4.6, it is also globally optimal.

Corollary 5.2. *Given G_c as the CG of the value function Q^π for all policies π. Let $\{\pi^k\}_{k=1}^\infty$ be the policy sequence generated by Algorithm 1 with an ADG G_d. If G_d is compatible with G_c, then $\{\pi^k\}_{k=1}^\infty$ converges to a globally optimal policy in a finite number of steps.*

In more general scenarios, the CG may be dynamic. Such changes in the CG are typically driven by both the state and the joint policy. For state-dependent changes, it suffices to design a distinct ADG for each state, ensuring that Corollary 4.6 remains valid. In contrast, for policy-dependent changes, the ADG should be updated after each policy iteration to preserve convergence to the globally optimal policy. A key challenge arises here: directly modifying the ADG may alter the structural dependencies of individual policies. For instance, under one ADG, agent i may depend only on the action of agent j, whereas under another ADG it may additionally depend on the actions of both j and k. If the policy of agent i lacks the interface to accommodate these additional dependencies, direct modification of the ADG becomes infeasible.

To address this issue, we could employ an indirect construction. Specifically, we build a set of individual policies such that, although their functional forms may differ before and after the ADG update, their induced joint policies remain identical. For deterministic policies, this construction is straightforward. Given a joint policy $\tilde{\pi}$ with independent components $\tilde{\pi}(s) = (\tilde{\pi}_1(s), \ldots, \tilde{\pi}_n(s))$, we define a set of action-dependent policies $\{\pi_i\}_{i\in\mathcal{N}}$ with the new ADG such that the ADG is compatible with the new CG $G_c(\tilde{\pi})$ and

$$\begin{aligned} &\pi_1(s) \leftarrow \tilde{\pi}_1(s), \quad \pi_2(s, \pi_1(s)) \leftarrow \tilde{\pi}_2(s), \\ &\ldots, \quad \pi_n(s, \tilde{\pi}_1(s), \ldots, \tilde{\pi}_{n-1}(s)) \leftarrow \tilde{\pi}_n(s). \end{aligned} \tag{13}$$

This guarantees that the joint policies before and after the ADG update are equivalent. Consequently, if the CG changes after policy improvement, we reconstruct the new policy π_i^{k+1} according to (13). This ensures that the ADG continue to be compatible with the CG, thereby allowing Algorithm 1 to achieve the optimal solution.

Corollary 5.3 (Proof in Appendix B)**.** *Given $G_c(\pi)$ as the CG of the value function Q^π for joint policy π. Let $\{\pi^k\}_{k=1}^\infty$ be the policy sequence generated by Algorithm 1, with policy reconstruction according to* (13) *upon CG change. Then $\{\pi^k\}_{k=1}^\infty$ converges to a globally optimal policy in a finite number of steps.*

6 EXPERIMENTS

6.1 COORDINATION POLYMATRIX GAMES

To validate theoretical optimality and convergence guarantees, we evaluate Algorithm 1 on polymatrix games with four CG topologies: star (5 agents), ring (5 agents), tree (7 agents), and mesh (9 agents), respectively. These topologies represent varying degrees of structural complexity. We compare three ADG configurations: sparse (constructed via Algorithm 2), fully dense (auto-regressive), and empty (independent). Additional experimental details are provided in Appendix D.

Figure 5 reports the learning curves averaged over 100 runs, showing that both sparse and dense ADGs consistently achieve the global reward upper bound across all topologies. In contrast, independent policies (empty ADGs) frequently trap in suboptimal Nash equilibria. Shaded areas represent the 95% confidence intervals (similarly hereafter) computed over 100 independent runs.

To assess computational efficiency, Figure 6 reports the average per-iteration runtime for different topologies, including ADGs derived from ring, star, and binary tree CGs, as well as fully dense ADGs. Each average time evaluation involves 100 runs, and each agent has two actions. It highlights the scalability advantage: while dense ADGs incur exponential runtime growth as the number of agents increases due to the enlarging joint action space, sparse ADGs exhibit resonable runtime while preserving the same performance.

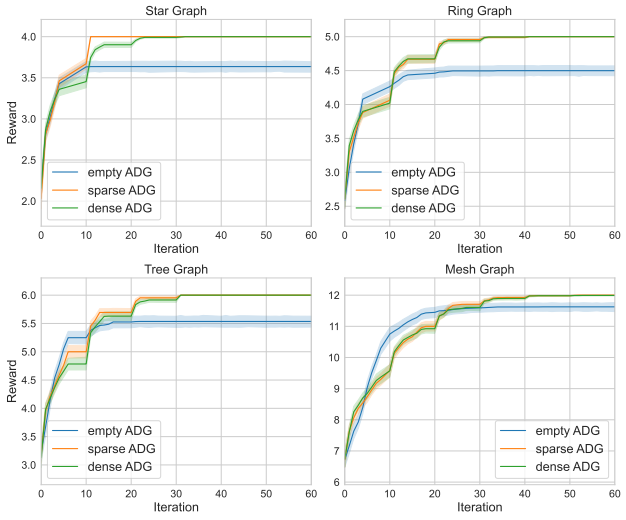

Figure 5: Results of coordination polymatrix game.

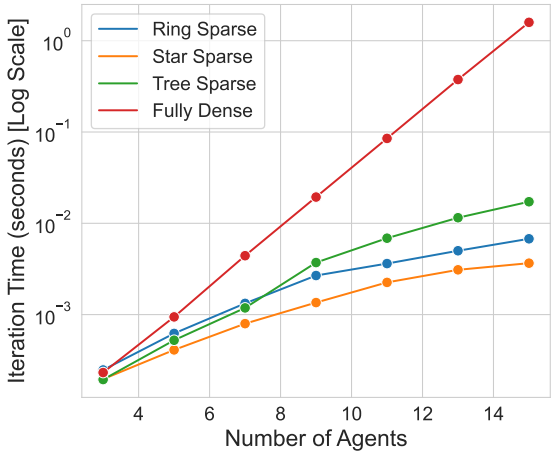

Figure 6: Average time per iteration.

6.2 IMPLEMENTATION IN DEEP MARL

To assess the performance of action-dependent policies with sparse ADGs in non-tabular settings, we integrate the ADG structure into the MAPPO framework [Yu et al., 2022]. Specifically, we replace the independent actor $\pi(a_i|s)$ with the action-dependent form $\pi(a_i|s, a_{N_d(i)})$ following the dependency structure defined by the ADG. This requires updating the PPO importance sampling ratio to account for the conditional action dependencies. Detailed experimental protocols and hyperparameters are provided in Appendix D.

Results on star-topology polymatrix games (lower right corner of Figure 7) demonstrate that even with neural network function approximation, ADG-structured policies could successfully escape suboptimal points that hinder independent learners. This confirms that the proposed structural conditions provide a robust design principle beyond tabular cases.

We further evaluate the framework on Adaptive Traffic Signal Control (ATSC) over 10 different seeds, where the coordination structure is naturally defined by the road network topology. We test on 2×2 (4 agents), 3×3 (9 agents), and 4×4 (16 agents) grid networks using the Simulation of Urban Mobility (SUMO) platform [Lopez et al., 2018] and SUMO-RL [Alegre, 2019]. The CG edges represent adjacency between intersections, and sparse ADGs are derived using greedy ordering in Algorithm 2.

Figure 7 shows that sparse ADGs achieve performance similar to auto-regressive models while outperforming independent policies. These results indicate that in realistic simulation tasks with CG structures, sparse ADGs can effectively capture necessary coordination without the computational overhead of auto-regressive communication.

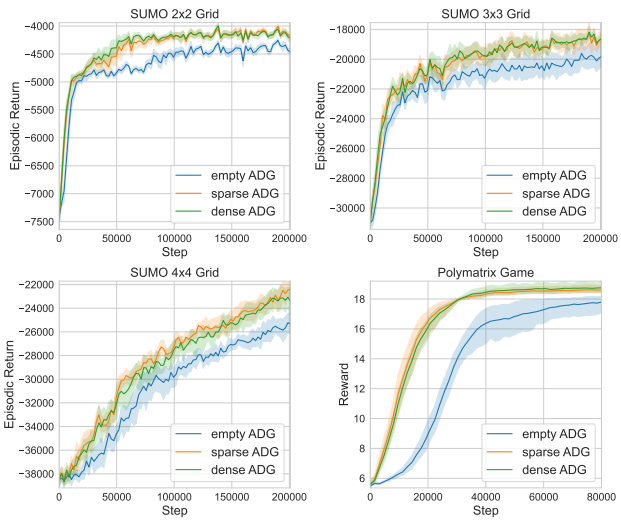

Figure 7: Results in Deep MARL.

7 CONCLUSION

In this work, we established a theoretical framework for action-dependent policies in multi-agent reinforcement learning by introducing the ADG and the G_d-local optimality. We further identified the conditions under which G_d-locally optimal policies coincide with globally optimal solutions in CG-structured problems, and proposed a policy iteration algorithm with guaranteed convergence. These theoretical findings were substantiated by empirical evaluations on polymatrix games and adaptive traffic signal control, confirming that our theory serves as a reliable design principle for scalable multi-agent systems. Future research will focus on developing automated methods for structural learning in environments where the underlying coordination topology is unknown.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (grants 62573068 and 62533002). The authors would like to thank the anonymous reviewers for their valuable comments.

References

Lucas N. Alegre. SUMO-RL. `https://github.com/LucasAlegre/sumo-rl`, 2019.

Eugenio Bargiacchi, Timothy Verstraeten, Diederik Roijers, Ann Nowé, and Hado Hasselt. Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision problems. In *International Conference on Machine Learning*, pages 482–490, 2018.

Umberto Bertele and Francesco Brioschi. *Nonserial dynamic programming*. Academic Press, Inc., 1972.

Dimitri Bertsekas. Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA Journal of Automatica Sinica*, 8(2):249–272, 2021.

Ashmita Bhattacharya and Malyaban Bal. Multi-Agent decision S4: Leveraging state space models for offline Multi-Agent reinforcement learning, 2025.

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

Maxime Bouton, Hasan Farooq, Julien Forgeat, Shruti Bothe, Meral Shirazipour, and Per Karlsson. Coordinated reinforcement learning for optimizing mobile networks. *arXiv preprint arXiv:2109.15175*, 2021.

Yang Cai and Constantinos Daskalakis. On minmax theorems for multiplayer games. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 217–234, 2011.

Jacopo Castellini, Frans A Oliehoek, Rahul Savani, and Shimon Whiteson. Analysing factorizations of action-value networks for cooperative multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 35(2):25, 2021.

Dingyang Chen and Qi Zhang. Context-aware Bayesian network actor-critic methods for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5327–5350, 2023.

Dingyang Chen, Zhenyu Zhang, Xiaolong Kuang, Xinyang Shen, Ozalp Ozer, and Qi Zhang. Convergence rates of Bayesian network policy gradient for cooperative multi-agent reinforcement learning. In *Neural Information Processing Systems Workshop on Bayesian Decision-making and Uncertainty*, 2024.

Zehao Dou, Jakub Grudzien Kuba, and Yaodong Yang. Understanding value decomposition algorithms in deep cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2202.04868*, 2022.

Wei Duan, Jie Lu, and Junyu Xuan. Group-aware coordination graph for multi-agent reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pages 3926–3934, 2024.

Jakob 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.

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative Multi-Agent reinforcement learning. In *International Conference on Machine Learning*, pages 6863–6877, 2022.

Carlos Guestrin, Michail G Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *International Conference on Machine Learning*, pages 227–234, 2002.

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

Gangshan Jing, He Bai, Jemin George, Aranya Chakrabortty, and Piyush K. Sharma. Distributed multiagent reinforcement learning based on graph-induced local value functions. *IEEE Transactions on Automatic Control*, 69(10): 6636–6651, 2024.

Jelle R Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7(65):1789–1828, 2006.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

JG Kuba, R Chen, M Wen, Y Wen, F Sun, J Wang, and Y Yang. Trust region policy optimisation in Multi-Agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Chuming Li, Jie Liu, Yinmin Zhang, Yuhong Wei, Yazhe Niu, Yaodong Yang, Yu Liu, and Wanli Ouyang. ACE: Cooperative multi-agent Q-learning with bidirectional action-dependency. In *AAAI Conference on Artificial Intelligence*, pages 8536–8544, 2023.

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.

Zhiyuan Li, Wenshuai Zhao, Lijun Wu, and Joni Pajarinen. Backpropagation through agents. In *AAAI Conference on Artificial Intelligence*, pages 13718–13726, 2024.

Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using SUMO. In *IEEE International Conference on Intelligent Transportation Systems*, pages 2575–2582, 2018.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6382–6393, 2017.

Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722, 2023.

Liviu Panait, Sean Luke, and R Paul Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6): 629–645, 2006.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking Multi-Agent deep reinforcement learning algorithms in cooperative tasks. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep Multi-Agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Alex Rogers, Alessandro Farinelli, Ruben Stranders, and Nicholas R Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759, 2011.

Jingqing Ruan, Yali Du, Xuantang Xiong, Dengpeng Xing, Xiyun Li, Linghui Meng, Haifeng Zhang, Jun Wang, and Bo Xu. GCS: Graph-based coordination strategy for Multi-Agent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1128–1136, 2022.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl 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.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius 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.

Richard S Sutton. *Reinforcement learning: An introduction*. MIT press, 2018.

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

Jingcheng Tang, Peng Zhou, He Bai, and Gangshan Jing. GRA: Graph-based reward aggregation for cooperative multi-agent reinforcement learning. *Journal of Automation and Intelligence*, 5(1):46–56, 2025.

Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-Agent conditional policy factorization. In *International Conference on Learning Representations*, 2022a.

Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. In *International Conference on Learning Representations*, 2022b.

Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, pages 16509–16521, 2022.

Jianing Ye, Chenghao Li, Jianhao Wang, Qianchuan Zhao, and Chongjie Zhang. Towards global optimality in cooperative MARL with sequential transformation, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, pages 24611–24624, 2022.

Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *AAAI Conference on Artificial Intelligence*, pages 764–770, 2011.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

Runyu Zhang, Jincheng Mei, Bo Dai, Dale Schuurmans, and Na Li. On the global convergence rates of decentralized softmax gradient play in Markov potential games. In *Advances in Neural Information Processing Systems*, pages 1923–1935, 2022.

Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Yanhao Huang, Jie Song, and Mingli Song. Is centralized training with decentralized execution framework centralized enough for MARL? *arXiv preprint arXiv:2305.17352*, 2023.

Sparse Action-Dependent Policy Iteration: Convergence and Optimality under Coordination Structures (Supplementary Material)

Jianglin Ding[1] **Jingcheng Tang**[1] **Gangshan Jing**[1]

[1]School of Automation, Chongqing University, Chongqing, China

A PROOF OF OPTIMALITY

Lemma A.1. *Let G_b be a DAG without immoralities under the topological order $1, \dots, n$. If a function $g_i(a_{\leq i})$ decomposes over the subgraph $G_{b,\leq i}$ as $g_i(a_{\leq i}) = \sum_{j=1}^{i} f_j(a_{N_b(j)}, a_j)$, then its maximization over a_i strictly preserves the decomposition over the remaining subgraph $G_{b,<i}$:*

$$\max_{a_i \in \mathcal{A}_i} g_i(a_{\leq i}) = \sum_{j=1}^{i-1} f'_j(a_{N_b(j)}, a_j), \tag{14}$$

where f'_j are updated local functions.

Proof. We isolate the terms involving a_i in $g_i(a_{\leq i})$:

$$\max_{a_i \in \mathcal{A}_i} g_i(a_{\leq i}) = \sum_{j=1}^{i-1} f_j(a_{N_b(j)}, a_j) + \max_{a_i \in \mathcal{A}_i} f_i(a_{N_b(i)}, a_i). \tag{15}$$

If $N_b(i) = \emptyset$, $\max_{a_i \in \mathcal{A}_i} f_i(a_i)$ is a constant and can be trivially added to any remaining function. Otherwise, let $k = \max(N_b(i))$ be the highest-indexed parent of i. Since G_b has no immoralities, all nodes in $N_b(i)$ are fully connected. Thus, nodes in $N_b(i) \setminus \{k\}$ must strictly precede k and be connected to k, which implies $N_b(i) \setminus \{k\} \subseteq N_b(k)$.

Consequently, the variables involved in $\max_{a_i \in \mathcal{A}_i} f_i(a_{N_b(i)}, a_i)$ are exactly a subset of $\{a_k\} \cup a_{N_b(k)}$. We can thus absorb this maximization term directly into f_k by defining:

$$f'_j(a_{N_b(j)}, a_j) = \begin{cases} f_k(a_{N_b(k)}, a_k) + \max_{a_i \in \mathcal{A}_i} f_i(a_{N_b(i)}, a_i), & \text{if } j = k, \\ f_j(a_{N_b(j)}, a_j), & \text{otherwise.} \end{cases} \tag{16}$$

Substituting f'_j back into (15) yields $\max_{a_i \in \mathcal{A}_i} g_i(a_{\leq i}) = \sum_{j=1}^{i-1} f'_j(a_{N_b(j)}, a_j)$. □

Proof of Theorem 4.4

Proof. Without loss of generality, we assume the agent indices $1, \dots, n$ follow a topological order of the DAG G_b. Fix an arbitrary state $s \in \mathcal{S}$ and denote $f(a) := Q^\pi(s, a)$. Since G_d is compatible with f, there exists a DAG G_b without immoralities such that:

$$f(a) = \sum_{i=1}^{n} f_i(a_{N_b(i)}, a_i), \quad \text{and} \quad N_b(i) \subseteq N_d(i),\ \forall i \in \mathcal{N}. \tag{17}$$

*Correspondence to Gangshan Jing: jinggangshan@cqu.edu.cn

According to the G_d-local optimality condition (10), for any agent i, the policy π_i unilaterally maximizes the joint objective when $a_{N_d(i)}$ is fixed:

$$\pi_i(a_{N_d(i)}) \in \arg\max_{a_i \in \mathcal{A}_i} f(\pi_{\mathcal{N}, N_d[i]}(a_{N_d[i]})). \tag{18}$$

We prove by backward induction on $i \in \{n, \dots, 0\}$ that for any prefix of fixed actions $a_{\leq i}$, the policy π rolls out optimally over the remaining agents:

$$f(\pi_{\mathcal{N}, \leq i}(a_{\leq i})) = \max_{a_{>i}} f(a_{\leq i}, a_{>i}). \tag{19}$$

Base case: Since all actions are fixed, $f(\pi_{\mathcal{N}, \leq n}(a_{\leq n})) = f(a) = \max_{a_{>n}} f(a_{\leq n}, a_{>n})$ holds trivially.

Inductive step: Assume (19) holds for agent i. We must show $f(\pi_{\mathcal{N}, <i}(a_{<i})) = \max_{a_{\geq i}} f(a_{<i}, a_{\geq i})$. By applying Lemma A.1, the function $\max_{a_{>i}} f(a_{\leq i}, a_{>i})$ preserves the additive structure over the remaining variables:

$$\max_{a_{>i}} f(a_{\leq i}, a_{>i}) = \sum_{j=1}^{i-1} f_j(a_{N_b(j)}, a_j) + f_i(a_{N_b(i)}, a_i). \tag{20}$$

Evaluating the objective in (18) fixes a_i and the parent actions $a_{N_d(i)}$. Let the preceding actions be $\tilde{a}_{<i} = \pi_{<i, N_d(i)}(a_{N_d(i)})$. By the induction hypothesis (19):

$$f(\pi_{\mathcal{N}, N_d[i]}(a_{N_d[i]})) = \max_{a_{>i}} f(\tilde{a}_{<i}, a_i, a_{>i}) = \sum_{j=1}^{i-1} f_j(\tilde{a}_{N_b(j)}, \tilde{a}_j) + f_i(\tilde{a}_{N_b(i)}, a_i). \tag{21}$$

Crucially, because compatibility guarantees $N_b(i) \subseteq N_d(i)$, the required parent actions for f_i are explicitly fixed in $a_{N_d(i)}$, implying $\tilde{a}_{N_b(i)} = a_{N_b(i)}$. Since the first summation in (21) is independent of a_i, substituting (21) into (18) yields:

$$\pi_i(a_{N_d(i)}) \in \arg\max_{a_i \in \mathcal{A}_i} f_i(a_{N_b(i)}, a_i). \tag{22}$$

We can now evaluate $f(\pi_{\mathcal{N}, <i}(a_{<i}))$:

$$\begin{aligned}
f(\pi_{\mathcal{N}, <i}(a_{<i})) &= f(\pi_{\mathcal{N}, \leq i}(a_{<i}, \pi_i(s, a_{<i}))) \\
&= \max_{a_{>i}} f(a_{<i}, \pi_i(s, a_{<i}), a_{>i}) \quad \text{(by the induction hypothesis)} \\
&= \sum_{j=1}^{i-1} f_j(a_{N_b(j)}, a_j) + f_i(a_{N_b(i)}, \pi_i(s, a_{<i})) \\
&= \sum_{j=1}^{i-1} f_j(a_{N_b(j)}, a_j) + \max_{a_i \in \mathcal{A}_i} f_i(a_{N_b(i)}, a_i) \quad \text{(due to (22))} \\
&= \max_{a_i \in \mathcal{A}_i} \left[\sum_{j=1}^{i-1} f_j(a_{N_b(j)}, a_j) + f_i(a_{N_b(i)}, a_i) \right] \\
&= \max_{a_i \in \mathcal{A}_i} \max_{a_{>i}} f(a_{\leq i}, a_{>i}) = \max_{a_{\geq i}} f(a_{<i}, a_{\geq i}).
\end{aligned} \tag{23}$$

By induction, reaching $i = 0$ yields $f(\pi(s)) = \max_a f(a)$. Recalling that $f(a) := Q^\pi(s, a)$, we have

$$V^\pi(s) = \mathcal{T}_\pi V^\pi(s) = Q^\pi(s, \pi(s)) = \max_{a \in \mathcal{A}} Q^\pi(s, a) = \mathcal{T} V^\pi(s), \tag{24}$$

which shows that V^π is a fixed point of $\mathcal{T}$. Hence, π is globally optimal. □

Proof of Corollary 4.6

Proof. By the definition of CG, since G_h is a chordal completion of G_c (i.e., $G_c \subseteq G_h$), $Q^\pi(s, a)$ trivially decomposes over G_h.

In probabilistic graphical models, the additive decomposition of a real-valued function is equivalent to the multiplicative factorization of a positive probability distribution (e.g., via the energy function $P(a) \propto \exp(Q^\pi(s,a))$). Thus, G_h acts as a Markov network for this distribution.

Since G_h is a chordal Markov network, Theorem 4.13 in Koller and Friedman [2009] guarantees the existence of a Bayesian network G_b that is a perfect map of G_h. The properties of a perfect map imply that G_b and G_h share the exact same underlying edge set, and G_b contains no immoralities.

Because the distribution factorizes over the Bayesian network G_b, taking the logarithm yields an exact additive sequential decomposition of $Q^\pi(s,a)$ over G_b:

$$Q^\pi(s,a) = \sum_{i \in \mathcal{N}} f_i(a_{N_b(i)}, a_i). \tag{25}$$

Therefore, $Q^\pi(s,a)$ additively decomposes according to G_b. Since the corollary assumes $G_b \subseteq G_d$, it immediately follows from Definition 4.3 that G_d is compatible with $Q^\pi(s,\cdot)$ for all states $s \in \mathcal{S}$. Given that the policy π is G_d-locally optimal, Theorem 4.4 confirms that π is globally optimal. □

B PROOF OF CONVERGENCE

We first reformulate the MAMDP as a sequentially expanded MDP (SEMDP) [Ye et al., 2022, Li et al., 2023, Bhattacharya and Bal, 2025], and then establish our proofs on this reformulation. The SEMDP transforms a multi-agent system into a single-agent MDP by expanding the state space and it does not introduce any new assumptions. Our results can still be proven directly under the MAMDP formulation. However, introducing the SEMDP can greatly simplifies the notation used in the proofs.

Given an MAMDP $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \tilde{P}, \tilde{r}, \gamma^n \rangle$, we construct a SEMDP denoted by $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where

- $\mathcal{S}$ is identical to the state space in the MAMDP;
- $\mathcal{A} = \prod_{i=1}^n \mathcal{A}_i$ is identical to the joint action space in the MAMDP;
- $\mathcal{Z} = \bigcup_{i \in \mathcal{N}} \mathcal{Z}_i$ is the expanded state space, where $\mathcal{Z}_i = \mathcal{S} \times \prod_{j=1}^{i-1} \mathcal{A}_j$ is the individual expanded state space for agent i;
- $P : \mathcal{Z} \times \mathcal{Z} \times \bigcup_{i \in \mathcal{N}} \mathcal{A}_i \to [0,1]$ is the transition kernel of the SEMDP;
- $r : \bigcup_{i \in \mathcal{N}} (\mathcal{Z}_i \times \mathcal{A}_i) \to \mathbb{R}$ is the reward function of the SEMDP;

The SEMDP transition kernel is defined as

$$P((s, a_{\leq i}) | (s, a_{<i}), a_i) = 1, \quad \forall s \in \mathcal{S}, a_{\leq i} \in \mathcal{A}_{\leq i}, 0 < i < n, \tag{26}$$

$$P(s' | (s, a_{<n}), a_n) = \tilde{P}(s' | s, a), \quad \forall s \in \mathcal{S}, a_{\leq n} \in \mathcal{A}_{\leq n}. \tag{27}$$

The reward function is defined as

$$r((s, a_{<i}); a_i) = 0, \quad \forall s \in \mathcal{S}, a_{\leq i} \in \mathcal{A}_{\leq i}, 0 < i < n, \tag{28}$$

$$r((s, a_{<n}); a_n) = \tilde{r}(s, a), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{29}$$

In the appendix, the joint policy in the SEMDP is denoted by $\pi : \mathcal{Z} \to \bigcup_{i \in \mathcal{N}} \mathcal{A}$, where

$$\pi(s, a_{<i}) \in \mathcal{A}_i, \quad \forall s \in \mathcal{S}, a_{<i} \in \mathcal{A}_{<i}.$$

The individual policy of agent i is defined as $\pi_i := \pi|_{\mathcal{Z}_i}$, which restricts the function to $\mathcal{Z}_i$. This formulation is identical to the individual policy in the MAMDP. To distinguish between the joint policies in MAMDP and SEMDP, we rewrite the joint policy in the MAMDP as $\hat{\pi} := \pi^k_{\mathcal{N},\varnothing} = (\pi_{1,\varnothing}, \ldots, \pi_{n,\varnothing})$.

In the SEMDP, the state value function takes the form $V : \mathcal{Z} \to \mathbb{R}$, and the state-action value function takes the form $Q : \bigcup_{i \in \mathcal{N}} (\mathcal{Z}_i \times \mathcal{A}_i) \to \mathbb{R}$. For any $V \in \mathcal{R}(\mathcal{Z})$, we define

$$Q^V(s, a_{<i}; a_i) := r(s, a_{<i}; a_i) + \gamma \sum_{z' \in \mathcal{Z}} P(z' \mid (s, a_{<i}), a_i) V(z'). \tag{30}$$

The Bellman operator $\mathcal{T}_\pi$ and the optimal Bellman operator $\mathcal{T}$ are given by

$$\mathcal{T}_\pi V(z) = Q^V(s, a_{<i}; \pi_i(s, a_{<i})), \quad \mathcal{T}V(z) = \max_{a_i} Q^V(s, a_{<i}; a_i). \tag{31}$$

The state value function V^π is the fixed point of $\mathcal{T}_\pi$, and the state-action value function is $Q^\pi := Q^{V^\pi}$. The optimal state value function V^* is the fixed point of $\mathcal{T}$, and $Q^* := Q^{V^*}$. The semicolons in Q and r highlight the action dependence. However, $Q(s, a_{<n}; a_n)$ can also be regarded as the value function of the full joint action and thus is sometimes written simply as $Q(s, a)$.

Proposition B.1. *We summarize several fundamental properties of the SEMDP:*

1. $V^{\hat{\pi}}(s) = \gamma^{1-n} V^\pi(s)$, $Q^{\hat{\pi}}(s, a) = \gamma^{1-n} Q^\pi(s, a)$, *where $V^{\hat{\pi}}$ and $Q^{\hat{\pi}}$ denotes the state and state-action value function in the original MAMDP.*
2. $Q^\pi(s, a_{<i}; a_i) = \gamma V^\pi(s, a_{\le i}), \quad 1 \le i < n.$
3. $Q^\pi(s, a_{<i}; a_i) = \gamma^{n-i} Q^\pi(s, \pi_{(<n),(\le i)}(s, a_{\le i}); \pi_{n,(\le i)}(s, a_{\le i})), \quad 1 \le i < n.$

Proof. (i) We expand the definition of $V^{\hat{\pi}}$ in the MAMDP:

$$\begin{aligned}
V^{\hat{\pi}}(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{nt} r(s_t, \hat{\pi}(s_t)) \middle| s_{t+1} \sim P(\cdot|s_t, \hat{\pi}(s_t)), s_0 = s\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \sum_{i=1}^{n} \gamma^{nt+i-n} r(s_t, \pi_{<i,\varnothing}(s_t); \pi_{i,\varnothing}(s_t)) \middle| s_{t+1} \sim P(\cdot|s_t, \hat{\pi}(s_t)), s_0 = s\right] \\
&= \mathbb{E}\left[\gamma^{1-n} \sum_{t=0}^{\infty} \sum_{i=1}^{n} \gamma^{nt+i-1} r(s_t, \pi_{<i,\varnothing}(s_t); \pi_{i,\varnothing}(s_t)) \middle| s_{t+1} \sim P(\cdot|s_t, \hat{\pi}(s_t)), s_0 = s\right] \\
&= \gamma^{1-n} \mathbb{E}\left[\sum_{t'=0}^{\infty} \gamma^{t'} r(z_{t'}; \pi(z_{t'})) \middle| z_{t'+1} \sim P(\cdot|z_{t'}, \pi(z_{t'})), z_0 = s\right] \\
&= \gamma^{1-n} V^\pi(s)
\end{aligned}$$

Similarly, we can derive that $Q^{\hat{\pi}}(s) = \gamma^{1-n} Q^\pi(s)$.

(ii) For $1 \le i < n$, we compute:

$$\begin{aligned}
Q^\pi(s, a_{<i}; a_i) &= r(s, a_{<i}; a_i) + \gamma \sum_{z' \in \mathcal{Z}} P(z' \mid (s, a_{<i}), a_i) V^\pi(z') \\
&= \gamma V^\pi(s, a_{\le i}).
\end{aligned}$$

(iii) Repeatedly unrolling the recursion yields:

$$\begin{aligned}
Q^\pi(s, a_{<i}; a_i) &= \gamma V^\pi(s, a_{\le i}) \\
&= \gamma Q^\pi(s, a_{\le i}; \pi_{i+1,(\le i)}(s, a_{\le i})) \\
&= \cdots = \gamma^{n-i} Q^\pi(s, \pi_{(<n),(\le i)}(s, a_{\le i}); \pi_{n,(\le i)}(s, a_{\le i})).
\end{aligned}$$

□

From Proposition B.1(i), we see that the value functions in the SEMDP and the MAMDP are equivalent up to a constant factor. Therefore, it can be easily verified that (1) an optimal policy in the SEMDP remains optimal in the original MAMDP, (2) Q^π and $Q^{\hat{\pi}}$ have the same CG, and (3) in (12), replacing the Q-function with its SEMDP counterpart does not affect the update outcome:

$$\pi_i^{k+1}(s, a_{<i}) \leftarrow \arg\max_{a_i} Q^{\pi^k}(s, \pi^{k+1}_{(<i),N_d(i)}(s, a_{N_d(i)}); a_i). \tag{32}$$

We first prove that the joint policy $\pi^k_{\mathcal{N},\varnothing}(s)$ in MAMDP converges.

Lemma B.2. *Let $\{\pi^k\}_{k=1}^{\infty}$ be the policy sequence generated by Algorithm 1. Then both $V^{\pi^k}(s)$ and $\pi^k_{\mathcal{N},\varnothing}(s)$ converge within a finite number of steps.*

Proof. From the update rule (32), for any $s \in \mathcal{S}, 0 \le i \le n$, we have

$$\begin{aligned}\mathcal{T}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)) &= Q^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s); \pi_i^{k+1}(s, \pi^{k+1}_{<i,\varnothing}(s))) \\ &\ge Q^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s); \pi_i^{k}(s, \pi^{k+1}_{<i,\varnothing}(s))) \\ &= V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)).\end{aligned} \tag{33}$$

We now proceed by induction. Assume that

$$\mathcal{T}^j_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)) \ge \mathcal{T}^{j-1}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)), \quad \forall s \in \mathcal{S},\ 0 \le i \le n. \tag{34}$$

We now prove that

$$\mathcal{T}^{j+1}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)) \ge \mathcal{T}^{j}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)), \quad \forall s \in \mathcal{S},\ 0 \le i \le n. \tag{35}$$

When $i < n$,

$$\begin{aligned}\mathcal{T}^{j+1}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)) &= \gamma \mathcal{T}^{j}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i+1,\varnothing}(s)) \\ &\ge \gamma \mathcal{T}^{j-1}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i+1,\varnothing}(s)) \\ &= \mathcal{T}^{j}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)).\end{aligned} \tag{36}$$

When $i = n$,

$$\begin{aligned}\mathcal{T}^{j+1}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<n,\varnothing}(s)) &= r(s, \pi^{k+1}_{\le n,\varnothing}(s)) + \gamma \sum_{s'} P(s'|s, \pi^{k+1}_{\le n,\varnothing}(s)) \mathcal{T}^{i}_{\pi^{k+1}} V^{\pi^k}(s') \\ &\ge r(s, \pi^{k+1}_{\le n,\varnothing}(s)) + \gamma \sum_{s'} P(s'|s, \pi^{k+1}_{\le n,\varnothing}(s)) \mathcal{T}^{i-1}_{\pi^{k+1}} V^{\pi^k}(s') \\ &= \mathcal{T}^{j}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<n,\varnothing}(s)).\end{aligned} \tag{37}$$

Thus,

$$V^{\pi^{k+1}}(s, \pi^{k+1}_{<i,\varnothing}(s)) = \lim_{j\to\infty} \mathcal{T}^{j}_{\pi^{k+1}} V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)) \ge V^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s)) \ge V^{\pi^k}(s, \pi^{k}_{<i,\varnothing}(s)). \tag{38}$$

By the monotone convergence theorem, $V^{\pi^k}(s, \pi^k_{<i,\varnothing}(s))$ converges. Since the policy space is finite, $V^{\pi^k}(s, \pi^k_{<i,\varnothing}(s))$, $\forall\, i \in \mathcal{N}, s \in \mathcal{S}$ converges within a finite number of steps.

Next, we prove by contradiction that $\pi^k_{\mathcal{N},\varnothing}(s)$ also converges. Suppose that for some M, $V^{\pi^k}(s)$ has already converged when $k \ge M$. Assume that for $k \ge M$, $\pi^k_{\mathcal{N},\varnothing}(s) \ne \pi^{k+1}_{\mathcal{N},\varnothing}(s)$. Let i be the first index such that $\pi^k_{i,\varnothing}(s) \ne \pi^{k+1}_{i,\varnothing}(s)$ while $\pi^k_{<i,\varnothing}(s) = \pi^{k+1}_{<i,\varnothing}(s)$. Then

$$\begin{aligned}V^{\pi^{k+1}}(s) &= \gamma V^{\pi^{k+1}}(s, \pi^{k+1}_{<2,\varnothing}(s)) = \cdots = \gamma^{n-1} V^{\pi^{k+1}}(s, \pi^{k+1}_{<n,\varnothing}(s)) \\ &= \gamma^{n-1}\left(r(s, \pi^{k+1}_{\le n,\varnothing}(s)) + \gamma \sum_{s'} P(s'|s, \pi^{k+1}_{\le n,\varnothing}(s)) V^{\pi^{k+1}}(s')\right) \\ &= \gamma^{n-1}\left(r(s, \pi^{k+1}_{\le n,\varnothing}(s)) + \gamma \sum_{s'} P(s'|s, \pi^{k+1}_{\le n,\varnothing}(s)) V^{\pi^{k}}(s')\right) \\ &= \gamma^{n-1} Q^{\pi^k}\left(s, \pi^{k+1}_{<n,\varnothing}(s); \pi_n^{k+1}(s, \pi^{k+1}_{<n,\varnothing}(s))\right) \\ &\ge \gamma^{n-1} Q^{\pi^k}\left(s, \pi^{k+1}_{<n,\varnothing}(s); \pi_n^{k}(s, \pi^{k+1}_{<n,\varnothing}(s))\right) \\ &= \gamma^{n-2} Q^{\pi^k}\left(s, \pi^{k+1}_{<n-1,\varnothing}(s); \pi_{n-1}^{k+1}(s, \pi^{k+1}_{<n-1,\varnothing}(s))\right) \\ &\ge \cdots \ge \gamma^{i-1} Q^{\pi^k}(s, \pi^{k+1}_{<i,\varnothing}(s); \pi_i^{k+1}(s, \pi^{k+1}_{<i,\varnothing}(s))) \\ &> \gamma^{i-1} Q^{\pi^k}(s, \pi^{k}_{<i,\varnothing}(s); \pi_i^{k}(s, \pi^{k}_{<i,\varnothing}(s))) \\ &= \gamma^{i-1} V^{\pi^k}(s, \pi^{k}_{<i,\varnothing}(s)) = V^{\pi^k}(s).\end{aligned} \tag{39}$$

The first equality follows from Proposition B.1 (ii). The third equality uses the fact that $V^{\pi^k}(s)$ has already converged. The strict inequality follows from the update rule, which preferentially selects the pre-update policy.

Hence $V^{\pi^{k+1}}(s) > V^{\pi^k}(s)$, contradicting the assumption that $V^{\pi^k}(s)$ has converged. Therefore, $\pi^k_{\mathcal{N},\varnothing}(s) = \pi^{k+1}_{\mathcal{N},\varnothing}(s)$ for $k \geq M$, implying that the joint policy $\pi^k_{\mathcal{N},\varnothing}(s)$ also converges. □

In order to deduce the convergence of individual policies from the convergence of $\pi^k_{\mathcal{N},\varnothing}(s)$, we employ induction. To make the induction work properly, we need to construct a special ordering.

Definition B.3. Let $\mathcal{C} = \mathcal{P}(\{1, 2, \ldots, n-1\})$, where $\mathcal{P}$ denotes the power set. We introduce a binary relation $<$ on $\mathcal{C}$ as follows:

$$A < B \Longleftrightarrow \min(A \setminus B) > \min(B \setminus A),\ A, B \in \mathcal{C}. \tag{40}$$

For the case involving the empty set, we define $\min \varnothing := n$.

Lemma B.4. *For any $A, B \in \mathcal{C}$, the following holds:*

$$A < B \Longleftrightarrow \min(A \Delta B) \in B \setminus A,$$

where Δ denotes the symmetric difference, i.e., $A \Delta B := (A \setminus B) \cup (B \setminus A)$.

Proof. We first prove the direction "⇐". Let $k = \min(A \Delta B) \in B \setminus A$. Since $A \setminus B \subseteq A \Delta B$, we have $\min(A \Delta B) \leq \min(A \setminus B)$. Moreover, because $k \in B \setminus A$, it follows that $\min(A \Delta B) = \min(B \setminus A)$. Since $(B \setminus A) \cap (A \setminus B) = \varnothing$, we obtain $\min(A \Delta B) < \min(A \setminus B)$. Thus, $\min(B \setminus A) < \min(A \setminus B)$, i.e., $A < B$.

Conversely, assume $A < B$. Then $\min(B \setminus A) < \min(A \setminus B)$. Hence,

$$\min(A \Delta B) = \min\{\min(B \setminus A), \min(A \setminus B)\} = \min(B \setminus A). \tag{41}$$

Therefore, $\min(A \Delta B) \in B \setminus A$. □

Proposition B.5. *The binary relation $<$ on $\mathcal{C}$ is a strict total order.*

Proof. Irreflexivity and asymmetry are immediate. We now prove that $<$ is connected and transitive.

(Connectedness): Let $A, B \in \mathcal{C}$ with $A \neq B$. We distinguish three cases:

(i) If $A \subsetneq B$, then $B \setminus A \neq \varnothing$. Hence $\min(B \setminus A) \leq n-1$ and $\min(A \setminus B) = n > \min(B \setminus A)$, so $A < B$.

(ii) If $B \subsetneq A$, by symmetry we obtain $B < A$.

(iii) If $A \nsubseteq B$ and $B \nsubseteq A$, then $\min(A \setminus B) \neq \min(B \setminus A)$. Thus, either $A < B$ or $B < A$.

(Transitivity): Let $A, B, C \in \mathcal{C}$ with $A < B$ and $B < C$.

If $A = \varnothing$, then clearly $A < C$. Otherwise, set $k_1 = \min(A \Delta B)$ and $k_2 = \min(B \Delta C)$. By Lemma B.4, we have $k_1 \in B \setminus A$ and $k_2 \in C \setminus B$. We analyze three cases:

(i) If $k_1 < k_2$, then $k_1 \notin B \Delta C$. Since $k_1 \in B$, it follows that $k_1 \in B \cap C$. Moreover, as $k_1 \in B \setminus A$, we have $k_1 \in C \setminus A$. Now, we only need to show $k_1 = \min(A \Delta C)$. For all $i < k_1$ with $i \in A \cup B$, we have $i \notin A \Delta B$, hence $i \in A \cap B$. Since $k_1 < k_2$, we also get $i \in A \cap B \cap C$, implying $i \notin A \Delta C$. Thus $k_1 = \min(A \Delta C)$, and by Lemma B.4, $A < C$.

(ii) If $k_2 < k_1$, then by symmetry, $k_2 = \min(A \Delta C)$ and $k_2 \in C \setminus A$. Hence $A < C$.

(iii) If $k_1 = k_2$, then $k_1 \in B \setminus A$ and $k_1 \in C \setminus B$ simultaneously, which is a contradiction. Thus this case cannot occur.

Therefore, $<$ is transitive. □

After defining the strict total order $<$, we arrange the elements of $\mathcal{C}$ in ascending order as $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$.

Lemma B.6. *Let $C_m \in \mathcal{C}$ and $C_m \neq \varnothing$. Let G_d be the ADG of π under state s. If $N_d(i) \not\supseteq C_m, i \in \mathcal{N}$, denote $k = \min(C_m \setminus N_d(i))$, and define*

$$C_j = A \cup B := \{x \in C_m \mid x < k\} \cup \{x \in \{1, \ldots, n-1\} \mid x > k\}. \tag{42}$$

Then we have $j < m$. Furthermore, if $i \neq k$, the following holds:

$$\pi_{i,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) = \pi_{i,C_m}(s, a_{C_m}). \tag{43}$$

Proof. We first verify that $j < m$. Since $N_d(i) \not\supseteq C_m$, we have $C_m \setminus N_d(i) \neq \varnothing$, and hence $k < n$. By construction of C_j, $\min(C_m \setminus C_j) = k$. Therefore,

$$\min(C_j \Delta C_m) = \min\{\min(C_j \setminus C_m), \min(C_m \setminus C_j)\} = \min\{\min(C_j \setminus C_m), k\}. \tag{44}$$

Because $\min(C_j \setminus C_m) > k$, we obtain $\min(C_j \Delta C_m) = k \in C_m \setminus C_j$. By Lemma B.4, this implies $C_j < C_m$, i.e., $j < m$.

When $i \in C_m$, since $i \neq k$, we have $i \in C_m \cap C_j$, and therefore

$$\pi_{i,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) = a_i = \pi_{i,C_m}(s, a_{C_m}). \tag{45}$$

When $i \notin C_m$, we consider the two cases $k = 1$ and $k > 1$ respectively.

(i) $k = 1$.

In this case, $C_j = \{2, 3, \ldots, n-1\}$. We analyze the construction of $\pi_{i,C_j}(s, a_{C_j})$:

$$\begin{aligned}
\pi_{i,C_j}(s, a_{C_j}) =& \pi_i(s, \pi_{(<i),C_j}(s, a_{C_j})) \\
=& \pi_i(s, \pi_{1,C_j}(s, a_{C_j}), \pi_{(<i)\cap C_j,C_j}(s, a_{C_j})) \\
=& \pi_i(s, \pi_1(s), a_{(<i)\cap C_j}) \\
=& \pi_i(s, \pi_1(s), a_{(<i)\cap(C_m\setminus\{1\})}, a_{(<i)\setminus C_m}).
\end{aligned} \tag{46}$$

Substituting $\pi_{C_j,C_m}(s, a_{C_m})$ into a_{C_j}, we obtain

$$\begin{aligned}
&\pi_{i,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) \\
=& \pi_i(s, \pi_{(<i),C_j}(s, a_{C_j}))\big|_{a_{C_j}=\pi_{C_j,C_m}(s,a_{C_m})} \\
=& \pi_i(s, \pi_1(s), a_{(<i)\cap(C_m\setminus\{1\})}, \pi_{(<i)\setminus C_m,C_m}(s, a_{C_m})).
\end{aligned} \tag{47}$$

Since $1 \notin N_d(i)$, π_i does not depend on a_1. Thus, replacing $\pi_1(s)$ with a_1, we obtain

$$\begin{aligned}
&\pi_{i,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) \\
=& \pi_i(s, a_1, a_{(<i)\cap(C_m\setminus\{1\})}, \pi_{(<i)\setminus C_m,C_m}(s, a_{C_m})) \\
=& \pi_i(s, a_{(<i)\cap C_m}, \pi_{(<i)\setminus C_m,C_m}(s, a_{C_m})) \\
=& \pi_{i,C_m}(s, a_{C_m}).
\end{aligned} \tag{48}$$

(ii) $k > 1$.

We proceed by induction to prove that

$$\pi_{\ell,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) = \pi_{\ell,C_m}(s, a_{C_m}), \quad \forall\, 1 \leq \ell < k. \tag{49}$$

For $\ell = 1$, since $1 < k$, we have $1 \in C_m \cap C_j$. Thus,

$$\pi_{1,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) = a_1 = \pi_{1,C_m}(s, a_{C_m}). \tag{50}$$

Assume the statement (49) holds for all indices less than ℓ. For $\ell < k$, note that $(< \ell) \cap C_m = (< \ell) \cap C_j$ and $(< \ell) \setminus C_m = (< \ell) \setminus C_j$. Therefore,

$$\begin{aligned}
&\pi_{\ell,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m})) \\
=& \pi_\ell(s, a_{(<\ell)\cap C_j}, \pi_{(<\ell)\setminus C_j,C_j}(s, a_{C_j}))\big|_{a_{C_j}=\pi_{C_j,C_m}(s,a_{C_m})} \\
=& \pi_\ell(s, a_{(<\ell)\cap C_m}, \pi_{(<\ell)\setminus C_m,C_j}(s, \pi_{C_j,C_m}(s, a_{C_m}))).
\end{aligned} \tag{51}$$

By the induction hypothesis,

$$\pi_{(<\ell)\setminus C_m,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m}))=\pi_{(<\ell)\setminus C_m,C_m}(s,a_{C_m}), \tag{52}$$

which yields

$$\pi_{\ell,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m}))=\pi_\ell(s,a_{(<\ell)\cap C_m},\pi_{(<\ell)\setminus C_m,C_m}(s,a_{C_m}))=\pi_{\ell,C_m}(s,a_{C_m}). \tag{53}$$

Finally, similar to (47), analyzing the construction of $\pi_{i,C_j}(s,a_{C_j})$ gives

$$\begin{aligned}&\pi_{i,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m}))\\=&\pi_i(s,\pi_{<i,C_j}(s,a_{C_j}))\big|_{a_{C_j}=\pi_{C_j,C_m}(s,a_{C_m})}\\=&\pi_i(s,a_{(<i)\cap C_m},\pi_{(<k)\setminus C_m,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m})),\pi_{k,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m})),\pi_{C_j\setminus C_m,C_m}(s,a_{C_m})).\end{aligned} \tag{54}$$

Since $k\notin N_d(i)$, we can replace $\pi_{k,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m}))$ by $\pi_{k,C_m}(s,a_{C_m})$, yielding

$$\begin{aligned}&\pi_{i,C_j}(s,\pi_{C_j,C_m}(s,a_{C_m}))\\=&\pi_i(s,a_{(<i)\cap C_m},\pi_{(<k)\setminus C_m,C_m}(s,a_{C_m}),\pi_{k,C_m}(s,a_{C_m}),\pi_{C_j\setminus C_m,C_m}(s,a_{C_m}))\\=&\pi_i(s,a_{(<i)\cap C_m},\pi_{(<i)\setminus C_m,C_m}(s,a_{C_m}))\\=&\pi_{i,C_m}(s,a_{C_m}).\end{aligned} \tag{55}$$

□

Lemma B.7. *Let $\{\pi^k\}_{k=1}^\infty$ be the policy sequence generated by Algorithm 1. Then for every $C_m\in\mathcal{C}$, $\pi^k_{\mathcal{N},C_m}$ converges within a finite number of steps.*

Proof. We proceed by induction.

Base case: Consider $C_1=\varnothing$. By Lemma B.2, we directly obtain that $\pi^k_{\mathcal{N},C_1}$ converges in finitely many steps.

Induction step: Assume that $\pi^k_{\mathcal{N},C_j}$ has already converged for all $j<m$ when $k\geq M$. We now prove that $\pi^k_{\mathcal{N},C_m}$ also converges in finitely many steps.

We first show that, when $k\geq M$, for any $\max C_m<i\leq n$, the following inequality holds:

$$Q^{\pi^k}(s,\pi^{k+1}_{<i,C_m}(s,a_{C_m});\pi^{k+1}_{i,C_m}(s,a_{C_m}))\geq\gamma^{-1}Q^{\pi^k}(s,\pi^{k+1}_{<i-1,C_m}(s,a_{C_m});\pi^{k+1}_{i-1,C_m}(s,a_{C_m})). \tag{56}$$

(i): If $N_d(i)\supseteq C_m$, then by the update rule,

$$\begin{aligned}&Q^{\pi^k}(s,\pi^{k+1}_{<i,C_m}(s,a_{C_m});\pi^{k+1}_{i,C_m}(s,a_{C_m}))\\&=Q^{\pi^k}(s,\pi^{k+1}_{<i,C_m}(s,a_{C_m});\pi^{k+1}_i(s,\pi^{k+1}_{<i,C_m}(s,a_{C_m})))\\&\geq Q^{\pi^k}(s,\pi^{k+1}_{<i,C_m}(s,a_{C_m});\pi^{k}_i(s,\pi^{k+1}_{<i,C_m}(s,a_{C_m})))\\&=\gamma^{-1}Q^{\pi^k}(s,\pi^{k+1}_{<i-1,C_m}(s,a_{C_m});\pi^{k+1}_{i-1}(s,\pi^{k+1}_{<i-1,C_m}(s,a_{C_m}))).\end{aligned} \tag{57}$$

(ii): If $N_d(i)\not\supseteq C_m$, we then construct $C_j=\{x\in C_m\mid x<k'\}\cup\{x\in\{1,\dots,n-1\}\mid x>k'\}$ with $k'=\min(C_m\setminus N_d(i))$ according to Lemma B.6. Since $i>\max C_m$, we have $i\neq k'$, and therefore $\pi^k_{i,C_m}(s,a_{C_m})=\pi^k_{i,C_j}(s,\pi^k_{C_j,C_m}(s,a_{C_m}))$. By the induction hypothesis, $\pi^k_{\mathcal{N},C_j}$ has already converged. Hence,

$$\begin{aligned}\pi^{k+1}_{i,C_m}(s,a_{C_m})&=\pi^{k+1}_{i,C_j}(s,\pi^{k+1}_{C_j,C_m}(s,a_{C_m}))\\&=\pi^{k}_{i,C_j}(s,\pi^{k+1}_{C_j,C_m}(s,a_{C_m}))\\&=\pi^k_i(s,\pi^k_{<i,C_j}(s,a_{C_j}))\big|_{a_{C_j}=\pi^{k+1}_{C_j,C_m}(s,a_{C_m})}\\&=\pi^k_i(s,\pi^k_{<i,C_j}(s,\pi^{k+1}_{C_j,C_m}(s,a_{C_m}))).\end{aligned} \tag{58}$$

We examine each component of $\pi^{k+1}_{<i,C_j}(s, \pi^{k+1}_{C_j,C_m}(s, a_{C_m}))$. If $x \in (<i)$ and $x = k'$, since $k' \notin N_d(i)$, then π^k_i is independent of a_x, so we replace $\pi^k_{x,C_j}(s, \pi^{k+1}_{C_j,C_m}(s, a_{C_m}))$ with $\pi^{k+1}_{x,C_m}(s, a_{C_m})$ in (58). If $x \neq k'$, then we again apply Lemma B.6 and the induction hypothesis to obtain

$$\pi^k_{x,C_j}(s, \pi^{k+1}_{C_j,C_m}(s, a_{C_m})) = \pi^{k+1}_{x,C_j}(s, \pi^{k+1}_{C_j,C_m}(s, a_{C_m})) = \pi^{k+1}_{x,C_m}(s, a_{C_m}). \tag{59}$$

Thus,

$$\pi^{k+1}_{i,C_m}(s, a_{C_m}) = \pi^k_i(s, \pi^k_{<i,C_j}(s, \pi^{k+1}_{C_j,C_m}(s, a_{C_m}))) = \pi^k_i(s, \pi^{k+1}_{<i,C_m}(s, a_{C_m})). \tag{60}$$

Therefore,

$$\begin{aligned}
&Q^{\pi^k}(s, \pi^{k+1}_{<i,C_m}(s, a_{C_m}); \pi^{k+1}_{i,C_m}(s, a_{C_m})) \\
&= Q^{\pi^k}(s, \pi^{k+1}_{<i,C_m}(s, a_{C_m}); \pi^k_i(s, \pi^{k+1}_{<i,C_m}(s, a_{C_m}))) \\
&= \gamma^{-1} Q^{\pi^k}(s, \pi^{k+1}_{<i-1,C_m}(s, a_{C_m}); \pi^{k+1}_{i-1}(s, \pi^{k+1}_{<i-1,C_m}(s, a_{C_m}))).
\end{aligned} \tag{61}$$

Next, we prove

$$Q^{\pi^k}(s, \pi^{k+1}_{<\max C_m,C_m}(s, a_{C_m}); \pi^{k+1}_{\max C_m,C_m}(s, a_{C_m})) = Q^{\pi^k}(s, \pi^k_{<\max C_m,C_m}(s, a_{C_m}); \pi^k_{\max C_m,C_m}(s, a_{C_m})). \tag{62}$$

We examine each component of $\pi^{k+1}_{\leq \max C_m,C_m}(s, a_{C_m})$.

(iii): If $i \in C_m$, then $\pi^k_{i,C_m}(s, a_{C_m}) = a_i = \pi^{k+1}_{i,C_m}(s, a_{C_m})$.

(iv): If $i \notin C_m$, let $C_j = C_m \cap (<i)$. Since $i < \max C_m$, we have $C_j \subsetneq C_m$, and by Definition B.3, $j < m$. By the induction hypothesis, π^k_{i,C_j} has converged. As $\pi^k_{\leq i}$ depends only on the first $i-1$ agents, it follows that $\pi^k_{i,C_m} = \pi^k_{i,C_j}$. Therefore,

$$\pi^k_{i,C_m}(s, a_{C_m}) = \pi^k_{i,C_j}(s, a_{C_j}) = \pi^{k+1}_{i,C_j}(s, a_{C_j}) = \pi^{k+1}_{i,C_m}(s, a_{C_m}). \tag{63}$$

Thus, $\pi^{k+1}_{\leq \max C_m,C_m}(s, a_{C_m}) = \pi^k_{\leq \max C_m,C_m}(s, a_{C_m})$, and hence (62) holds.

Now,

$$\begin{aligned}
&Q^{\pi^{k+1}}(s, \pi^{k+1}_{<n,C_m}(s, a_{C_m}); \pi^{k+1}_{n,C_m}(s, a_{C_m})) \\
&= Q^{\pi^k}(s, \pi^{k+1}_{<n,C_m}(s, a_{C_m}); \pi^{k+1}_{n,C_m}(s, a_{C_m})) \\
&\geq \cdots \geq \gamma^{\max C_m - n} Q^{\pi^k}(s, \pi^{k+1}_{<\max C_m,C_m}(s, a_{C_m}); \pi^{k+1}_{\max C_m,C_m}(s, a_{C_m})) \\
&= \gamma^{\max C_m - n} Q^{\pi^k}(s, \pi^k_{<\max C_m,C_m}(s, a_{C_m}); \pi^k_{\max C_m,C_m}(s, a_{C_m})) \\
&= Q^{\pi^k}(s, \pi^k_{<n,C_m}(s, a_{C_m}); \pi^k_{n,C_m}(s, a_{C_m})).
\end{aligned} \tag{64}$$

Here, the second line follows from Lemma B.2, which ensures that $V^{\pi^k}(s)$ has converged, and hence $Q^{\pi^k}(s, a)$ have converged; the third line follows from (56); the fourth line from (62).

Equation (64) shows that $Q^{\pi^k}(s, \pi^k_{<n,C_m}(s, a_{C_m}); \pi^k_{n,C_m}(s, a_{C_m}))$ is monotonically non-decreasing, and thus converges in finitely many steps.

Let $k \geq M' \geq M$ be such that $Q^{\pi^k}(s, \pi^k_{<n,C_m}(s, a_{C_m}); \pi^k_{n,C_m}(s, a_{C_m}))$ has converged. We now prove by contradiction that $\pi^k_{\mathcal{N},C_m}$ must also converge.

Suppose for $k \geq M'$, $\pi^k_{\mathcal{N},C_m}(s) \neq \pi^{k+1}_{\mathcal{N},C_m}(s)$. Let i be the smallest index where they differ, i.e.,

$$\pi^k_{i,C_m}(s) \neq \pi^{k+1}_{i,C_m}(s), \quad \pi^k_{<i,C_m}(s) = \pi^{k+1}_{<i,C_m}(s). \tag{65}$$

By the analyses in (iii) and (iv), i must satisfy $i > \max C_m$. If $N_d(i) \not\supseteq C_m$, then by (ii),

$$\pi^{k+1}_{i,C_m}(s, a_{C_m}) = \pi^k_i(s, \pi^{k+1}_{<i,C_m}(s, a_{C_m})) = \pi^k_i(s, \pi^k_{<i,C_m}(s, a_{C_m})) = \pi^k_{i,C_m}(s, a_{C_m}), \tag{66}$$

contradicting $\pi^k_{i,C_m}(s) \neq \pi^{k+1}_{i,C_m}(s)$. Therefore, it must be that $N_d(i) \supseteq C_m$.

$$
\begin{aligned}
&Q^{\pi^{k+1}}(s, \pi^{k+1}_{<n,C_m}(s, a_{C_m}); \pi^{k+1}_{n,C_m}(s, a_{C_m})) \\
&= Q^{\pi^k}(s, \pi^{k+1}_{<n,C_m}(s, a_{C_m}); \pi^{k+1}_{n,C_m}(s, a_{C_m})) \\
&\geq \gamma^{i-n} Q^{\pi^k}(s, \pi^{k+1}_{<i,C_m}(s, a_{C_m}); \pi^{k+1}_{i,C_m}(s, a_{C_m})) \\
&= \gamma^{i-n} Q^{\pi^k}(s, \pi^{k}_{<i,C_m}(s, a_{C_m}); \pi^{k+1}_{i}(s, \pi^{k}_{<i,C_m}(s, a_{C_m}))) \\
&> \gamma^{i-n} Q^{\pi^k}(s, \pi^{k}_{<i,C_m}(s, a_{C_m}); \pi^{k}_{i}(s, \pi^{k}_{<i,C_m}(s, a_{C_m}))) \\
&= Q^{\pi^k}(s, \pi^{k}_{<n,C_m}(s, a_{C_m}); \pi^{k}_{n,C_m}(s, a_{C_m})).
\end{aligned} \tag{67}
$$

The second line uses the fact that $Q^{\pi^k}(s, a)$ has converged; the third line is by the update rule; the fifth line follows from the update rule, which preferentially selects the pre-update policy.

Equation (67) contradicts the fact that $Q^{\pi^k}(s, \pi^k_{<n,C_m}(s, a_{C_m}); \pi^k_{n,C_m}(s, a_{C_m}))$ has already converged. Therefore, for all $k \geq M'$, $\pi^k_{\mathcal{N},C_m}(s) = \pi^{k+1}_{\mathcal{N},C_m}(s)$, i.e., $\pi^k_{\mathcal{N},C_m}$ converges in finitely many steps. □

Proof of Theorem 5.1

Proof. According to Lemma B.7, we have that $\pi^k_i = \pi^k_{i,<i}$ converges in a finite number of steps. Let π be the limit point of the sequence $\{\pi^k\}_{k=1}^{\infty}$. From the update rule, it follows that

$$
Q^{\pi}(s, \pi_{(<i),N_d(i)}(s, a_{N_d(i)}); \pi_{i,N_d(i)}(s, a_{N_d(i)})) = \max_{a_i} Q^{\pi}(s, \pi_{(<i),N_d(i)}(s, a_{N_d(i)}); a_i). \tag{68}
$$

Therefore, the limit point of $\{\pi^k\}_{k=1}^{\infty}$ is a G_d-locally optimal policy. □

Proof of Corollary 5.3

Proof. From the analysis in Lemma B.2, we know that $V^{\pi^k}(s)$ is monotonically non-decreasing. Hence, $V^{\pi^k}(s)$ also converges within a finite number of steps.

Suppose that for $k \geq M$, $V^{\pi^k}(s)$ has already converged. Assume further that $G_c(\hat{\pi}^{k+1}) \neq G_c(\hat{\pi}^k)$ when $k \geq M$. This implies $\hat{\pi}^{k+1} \neq \hat{\pi}^k$. From the contradiction argument in Lemma B.2, it follows that

$$
V^{\pi^{k+1}}(s) = V^{\hat{\pi}^{k+1}}(s) > V^{\hat{\pi}^k}(s) = V^{\pi^k}(s), \tag{69}
$$

which contradicts the assumption that $V^{\pi^k}(s)$ has already converged. Therefore, it must hold that $G_c(\hat{\pi}^{k+1}) = G_c(\hat{\pi}^k)$ for all $k \geq M$.

Consequently, when $k \geq M$, the dynamic CG eventually stabilizes into a static CG. During this stabilization stage, the update process reduces to Algorithm 1. By Theorem 5.1, π^k converges within a finite number of steps. Let π be the limit point of $\{\pi^k\}_{k=1}^{\infty}$. Then π is a G_d-locally optimal policy. Furthermore, since the ADG is compatible with $G_c(\pi)$, it follows from Corollary 4.6 that π is globally optimal. □

C CONSTRUCTION OF ACTION DEPENDENCY GRAPHS

To elucidate the construction of ADGs, we present Algorithm 2 that efficiently derives an sparse ADG from a given CG.

D EXPERIMENTAL DETAILS

Setup of Coordination Polymatrix Game. In matrix cooperative games, different CGs and their corresponding sparse ADGs are shown in Figure 8. For policy iteration methods, we randomly generate the parameters of the payoff matrices, while fixing the maximum reward to be equal to the number of edges in the CG. Moreover, the maximum reward is obtained when all agents choose action 1. For the MAPPO method, we use fixed payoff matrices, with the exact parameters provided in Table 1 to Table 4.

Algorithm 2 Greedy Algorithm: Finding a Sparse ADG

Input: A Coordination Graph $G_c = (\mathcal{N}, \mathcal{E}_c)$
Output: A topological ordering of agents $\pi : \mathcal{N} \to \{1, \ldots, n\}$ and an ADG $G_d = (\mathcal{N}, \mathcal{E}_d)$
Initialize working graph $G_h \leftarrow G_c$, unmarked vertices $\mathcal{U} \leftarrow \mathcal{N}$, and ADG edges $\mathcal{E}_d \leftarrow \emptyset$
for $k = n$ **down to** 1 **do**
  for each $u \in \mathcal{U}$ **do**
    $N_h(u) \leftarrow \{w \in G_h \mid (u, w) \in G_h\}$
    $cost(u) \leftarrow$ number of missing edges between pairs of vertices in $N_h(u)$
  end for
  Find $v^* \in \arg\min_{u \in \mathcal{U}} cost(u)$
  Assign topological index $\pi(v^*) \leftarrow k$
  for each $w \in N_h(v^*)$ **do**
    $\mathcal{E}_d \leftarrow \mathcal{E}_d \cup \{(w, v^*)\}$
  end for
  Add undirected edges to G_h to fully connect $N_h(v^*)$ {Chordal completion to avoid immoralities}
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v^*\}$ {Mark v^* as eliminated}
end for
return ordering π and $G_d = (\mathcal{N}, \mathcal{E}_d)$

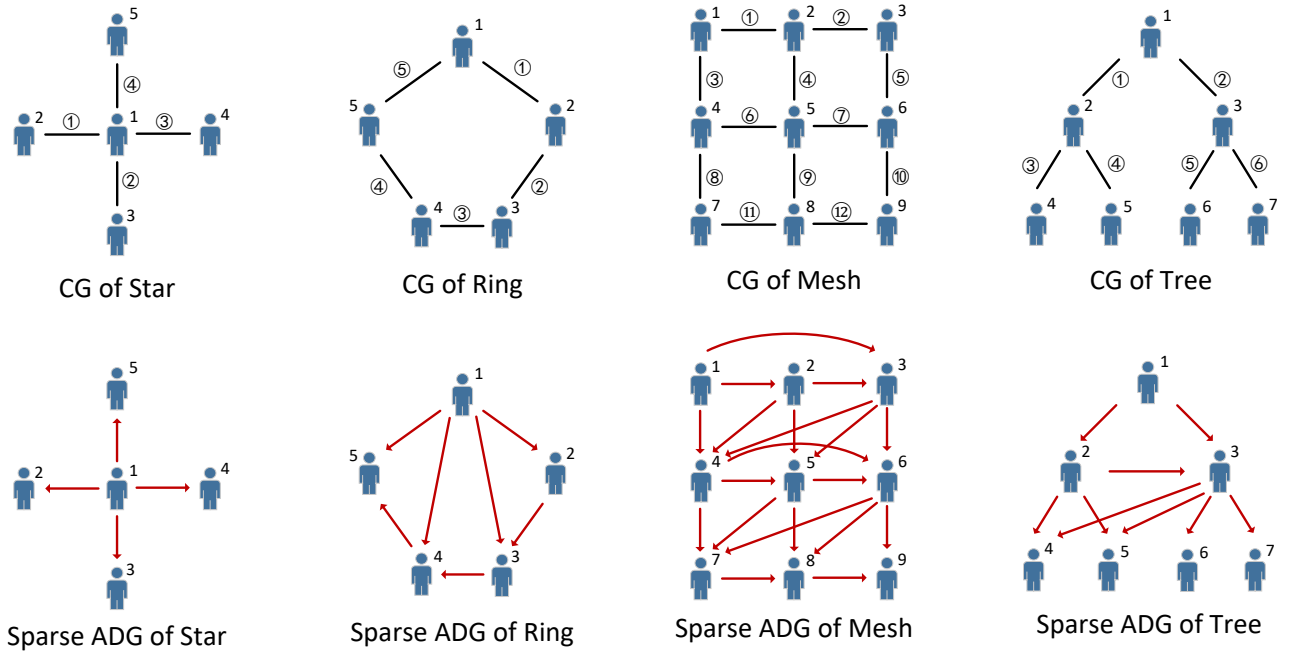

Figure 8: The CG and sparse ADG of polymatrix coordination game.

Table 1: ① in Star

$a_1 \backslash a_2$	0	1	2	3	4
0	3.5	5.0	0.5	0.5	0.5
1	0.5	3.5	6.0	0.5	0.5
2	0.5	0.5	3.5	0.5	0.5
3	0.5	0.5	0.5	3.25	0.5
4	0.5	0.5	0.5	0.5	3.0

Table 2: ② in Star

$a_1 \backslash a_3$	0	1	2	3	4
0	3.5	0.5	5.0	0.5	0.5
1	0.5	3.5	6.0	0.5	0.5
2	0.5	0.5	3.5	0.5	0.5
3	0.5	0.5	0.5	3.25	0.5
4	0.5	0.5	0.5	0.5	3.0

Table 3: ③ in Star

$a_1 \backslash a_4$	0	1	2	3	4
0	3.5	0.5	0.5	5.0	0.5
1	0.5	3.5	6.0	0.5	0.5
2	0.5	0.5	3.5	0.5	0.5
3	0.5	0.5	0.5	3.25	0.5
4	0.5	0.5	0.5	0.5	3.0

Table 4: ④ in Star

$a_1 \backslash a_5$	0	1	2	3	4
0	3.5	0.5	0.5	0.5	5.0
1	0.0	0.0	0.0	0.0	0.0
2	0.5	0.5	3.5	0.5	0.5
3	0.5	0.5	0.5	3.25	0.5
4	0.5	0.5	0.5	0.5	3.0

Setup of ATSC. In the ATSC environment, we conduct experiments on the maps `2x2grid`, `3x3grid`, and `RESCO/grid4x4` provided by SUMO-RL. Based on the road network connectivity, we design corresponding CGs, with their adjacency lists for CGs and sparse ADGs reported in Table 5 to Table 10. To increase task difficulty and highlight the benefits of cooperation, we follow the idea of Li et al. [2021], Böhmer et al. [2020] to modify the reward function. Specifically, instead of assigning each agent its own queue reward, we redefine the reward as the minimum individual reward among its CG neighbors, thereby emphasizing the performance gap induced by cooperation.

Table 5: CG of 2x2grid

Vertex	**Neighbors**
0	1, 2
1	0, 3
2	0, 3
3	1, 2

Table 6: Sparse ADG of 2x2grid

Vertex	**Parent nodes**
0	
1	0
2	1, 0
3	2, 1

Table 7: CG of 3x3grid

Vertex	**Neighbors**
0	1, 3
1	0, 2, 4
2	1, 5
3	0, 4, 6
4	1, 3, 5, 7
5	2, 4, 8
6	3, 7
7	4, 6, 8
8	5, 7

Table 8: Sparse ADG of 3x3grid

Vertex	**Parent nodes**
0	
1	0
2	1, 0
3	2, 1, 0
4	3, 2, 1
5	4, 3, 2
6	5, 4, 3
7	6, 5, 4
8	7, 5

Setings of MAPPO. In MAPPO, We transform the independent policy $\pi_{\theta_i}(a_i|s)$ into the action-dependent form $\pi_{\theta_i}(a_i|s, a_{N_d(i)})$. This requires a modification of the optimization objective to properly handle the action-dependent policy. Consider MAPPO [Yu et al., 2022], where the original objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_{\theta_{\text{old}}}}\left[\sum_{i=1}^{n} \min\left(r_{\theta_i}(a_i, s)A_{\pi_{\theta_{\text{old}}}}(s, a),\ \text{clip}(r_{\theta_i}(a_i, s), 1 \pm \varepsilon)A_{\pi_{\theta_{\text{old}}}}(s, a)\right)\right],$$

where $r_{\theta_i}(a_i, s) = \frac{\pi_{\theta_i}(a_i|s)}{\pi_{\theta_{i,\text{old}}}(a_i|s)}$, and $\mathcal{D}$ denotes the distribution in the replay buffer. To adapt this objective for action-dependent policies, we replace $r_{\theta_i}(a_i, s)$ with $r_{\theta_i}(a_i, s, a_{N_d(i)}) = \frac{\pi_{\theta_i}(a_i|s, a_{N_d(i)})}{\pi_{\theta_{i,\text{old}}}(a_i|s, a_{N_d(i)})}$.

For MAPPO with empty ADGs, we adopt the default MLP configurations specified in Papoudakis et al. [2021], Böhmer et al. [2020]. For MAPPO with sparse and dense ADGs, we modify the agent network architecture as follows.

Let $o_i \in \mathbb{R}^{d_o}$ denote the observational features of agent i, and $a_i \in \mathbb{R}^{d_a}$ the action features of agent i. The first-layer hidden features are computed as:

$$h^1_{o_i} = \text{ReLU}(W_1 o_i + b_1), \quad h^1_{a_i} = \text{ReLU}(W_2 a_i + b_2),$$

where $W_1 \in \mathbb{R}^{64\times d_o}$, $W_2 \in \mathbb{R}^{64\times d_a}$ and $b_1, b_2 \in \mathbb{R}^{64}$ are the weights and biases, respectively.

Table 9: CG of 4x4grid

Vertex	Neighbors
0	1, 4
1	0, 2, 5
2	1, 3, 6
3	2, 7
4	0, 5, 8
5	1, 4, 6, 9
6	2, 5, 7, 10
7	3, 6, 11
8	4, 9, 12
9	5, 8, 10, 13
10	6, 9, 11, 14
11	7, 10, 15
12	8, 13
13	9, 12, 14
14	10, 13, 15
15	11, 14

Table 10: Sparse ADG of 4x4grid

Vertex	Parent nodes
0	
1	0
2	0, 1
3	0, 1, 2
4	0, 1, 2, 3
5	1, 2, 3, 4
6	2, 3, 4, 5
7	3, 4, 5, 6
8	4, 5, 6, 7
9	5, 6, 7, 8
10	6, 7, 8, 9
11	7, 8, 9, 10
12	8, 9, 10, 11
13	9, 10, 11, 12
14	10, 11, 13
15	11, 14

Next, we take the average of $h^1_{a_i}$ over the dependency set $N_d(i)$:

$$h^2_{a_i} = \frac{1}{|N_d(i)|} \sum_{i \in N_d(i)} h^1_{a_i}.$$

We then concatenate the two feature vectors and obtain

$$h^3 = [h^1_{o_i}, h^2_{a_i}],$$

which is fed into a multilayer perceptron:

$$h^4 = \text{ReLU}(W_3 h^3 + b_3), \quad z = W_4 h^4 + b_4,$$

where $W_3 \in \mathbb{R}^{64\times 128}$, $W_4 \in \mathbb{R}^{d_{\text{action}}\times 64}$, $b_3 \in \mathbb{R}^{64}$, and $b_4 \in \mathbb{R}^{d_{\text{action}}}$. The final output is z.

Experimental Hyperparameters. Our implementation of MAPPO is based on the open-source EPyMARL framework Papoudakis et al. [2021], employing the Adam optimizer for training. We use the same hyperparameters across experiments under different ADGs, with a few critical hyperparameters adjusted to fit each environment. These modified values are reported in Table 11, while any unlisted parameters follow the default EPyMARL configuration.

Table 11: Experimental Hyperparameters

	learning rate	weight decay	buffer size	batch size	entropy coefficient
ATSC	0.0004	0.0001	8	8	0.02
Polymatrix Game	0.0004	0.0001	16	8	0.1

Extension to Unknown Coordination Structures. We further extended our approach to environments with unknown CGs by integrating it with online structure learning. Specifically, we conducted an experiment in the ALOHA environment [Wang et al., 2022b] where we do not use the prior CG knowledge. Instead, we employed the CASEC [Wang et al., 2022b] method to independently learn the coordination structure, which was then pruned to retain only the 30% most critical edges and converted into an ADG. To accommodate the variable number of dependent actions as the CG evolves during training, we utilized the Multi-Agent Transformer (MAT) [Wen et al., 2022] for policy training. MAT provides the flexibility to implement sparse action dependencies dynamically by applying an attention mask that blocks interactions outside the defined ADG structure.

All experiments were conducted across 10 different random seeds. Key hyperparameters for training included a learning rate of 0.0004, a buffer size and batch size of 10, and an entropy coefficient of 0.1. The experimental results (Figure 9)

comparing dense (auto-regressive), sparse (30% learned CG edges), and empty (independent) configurations show that the sparse ADG achieved performance identical to, and even slightly better than, the dense ADG, while both outperformed the empty baseline. This confirms that even when the CG learned from scratch, our theory still provides an effective and principled target for structural pruning.

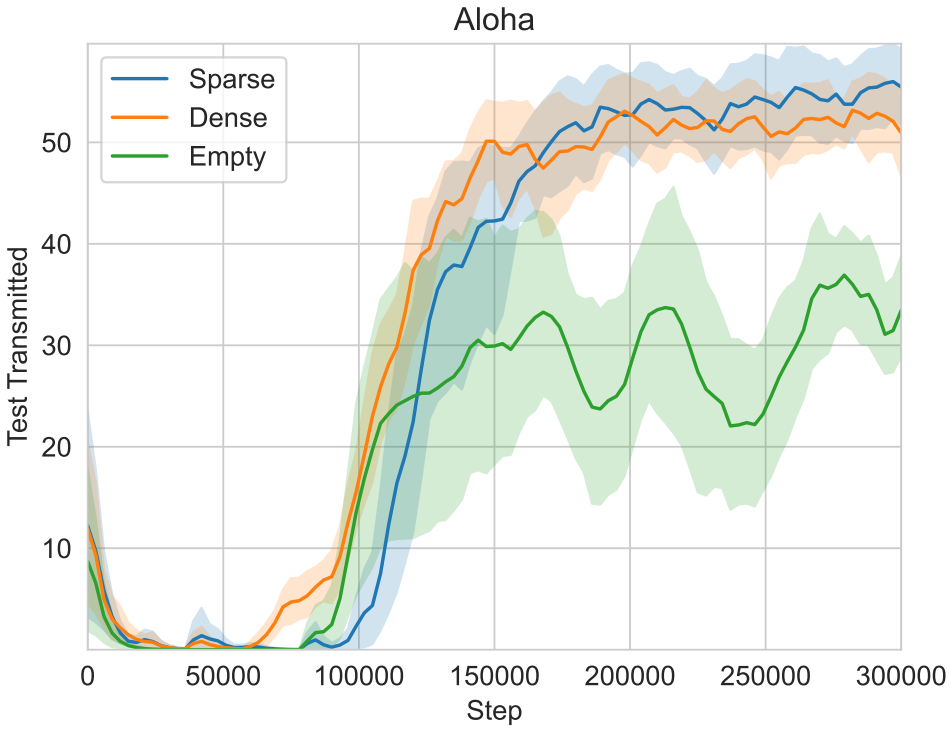

Figure 9: Results in ALOHA.