Shapley Meets DCOP: A Unified Structural Credit Assignment for Multiagent Planning and Multiagent Reinforcement Learning

Wanyuan Wang, *Member, IEEE*, Qian Che, Chunyu Liu, Youzhi Zhang, Jiuchuan Jiang, and Bo An, *Senior Member, IEEE*

***Abstract*—With the construction of intelligent agents, coordinating a collection of agents to optimize long-term cumulative global reward is becoming increasingly important. To align individual agent actions with global rewards, the contribution of individual actions to global rewards needs to be determined, which is known as the structural credit assignment (SCA). Conventional SCA mechanisms are primarily based on neural networks, which lack theoretical foundations and preclude their application to model-based MAP tasks. Leveraging cooperative game theory, the main contribution of this study is to propose a novel Shapley value-based SCA (SV-SCA) that can be generalized to both MAP and MARL. Combining the distributed constraint optimization (DCOP) model and its reward structure, we propose a novel algorithm for computing the Shapley value while ensuring the efficiency and fairness of the SV-SCA. Particularly, based on SV-SCA, we design a coordinated Monte Carlo tree search (MCTS) for model-based MAP tasks and a fully-decentralized method for model-free MARL tasks. Theoretical analyses show that the proposed coordinated MCTS can guarantee the expected value of the global joint action, and that the proposed coordinated MARL is monotonic such that each agent optimizes its own rewards also optimize the system's global reward. Finally, we conduct extensive experiments in typical sequential multiagent coordination domains. Our results demonstrate that the proposed coordinated MCTS and coordinated MARL outperform existing multiagent MCTS and MARL baselines in terms of solution quality and scalability.**

***Note to Practitioners*—This paper studies multiagent sequential decision making problems, where a team of agents needs to coordinate and take sequences of actions to optimize long-term overall rewards. Multiagent sequential decision making has been widely used in the filed of automation, such as traffic signal control, multi-robot/UAV control, and automatic vehicle coordination. Existing approaches, including multiagent planning (MAP) and multiagent reinforcement learning (MARL), fail to generalize the credit assignment problem of determining the contribution of individual actions to global rewards. This paper integrates the distributed constraint optimization model (a typical multiagent interaction structure that is ignored in traditional multiagent sequential decision making problems) and the Shapley value (a fair division of joint gains in cooperative game theory), and proposes a structural credit assignment (SCA) mechanism, which can be applied to both MAP and MARL. Based on the proposed SCA, we design a coordinated Monte Carlo tree search (MCTS) method for model-based MAP tasks (e.g., multi-robot coordination and traffic signal control) and a fully-decentralized model-free MARL methods (e.g., multi-rover search). In this paper, we mathematically show that the proposed coordinated MARL is monotonic, which provides a sufficient condition for convergence to the optimal global joint policy. Preliminary experiments suggest that the proposed coordinated MCTS and coordinated MARL can guarantee both the solution quality and scalability in various real-world domains. In future research, we will integrate coordinated MCTS and MARL into a unified framework for general multiagent decision-making.**

***Index Terms*—Multiagent planning, multiagent reinforcement learning, structure credit assignment, Shapley value, distributed constraint optimization (DCOP).**

Received 28 April 2025; revised 19 September 2025; accepted 27 October 2025. Date of publication 9 December 2025; date of current version 12 February 2026. This article was recommended for publication by Associate Editor B. Yan and Editor Q.-S. Jia upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2024YFB4303805, in part by the Key Research and Development Projects in Jiangsu Province under Grant BE2021001-2, and in part by the National Natural Science Foundation of China under Grant 62476121 and Grant 12201619. *(Corresponding author: Wanyuan Wang.)*

Wanyuan Wang and Chunyu Liu are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: wywang@seu.edu.cn; lcy88227176@163.com).

Qian Che is with Jiangsu Police College, Nanjing 210031, China (e-mail: qche@seu.edu.cn).

Youzhi Zhang is with the Centre for Artificial Intelligence and Robotics (CAIR), Hong Kong Institute of Science and Innovation, Hong Kong (e-mail: youzhi.zhang.lgc@gmail.com).

Jiuchuan Jiang is with Nanjing University of Finance and Economics, Nanjing 210023, China (e-mail: jcjiang@nufe.edu.cn).

Bo An is with Nanyang Technological University, Singapore 639798 (e-mail: boan@ntu.edu.sg).

Digital Object Identifier 10.1109/TASE.2025.3641970

I. Introduction

In COOPERATIVE multiagent systems (MASs), it is essential to coordinate a group of agents that take sequences of actions to optimize long-term overall rewards [1]. This sequential coordination problem can be formulated as a multiagent Markov decision process (MMDP), which has been widely used in the filed of automation, such as traffic signal control [2], [3], multi-robot/UAV manipulation [4], [5], [6], [7], [8], automatic vehicle coordination [9] and inventory control [10]. However, MMDP has been shown to be computationally intractable in general, and there are two main challenges: 1) the exponential combination action space, and 2) the exponential historical decision-making space [11].

In order to reduce the complexity of exponential combination action space, decentralized execution is essential, that is, to ensure that each agent performs its individual action while maximizing the global rewards. In this paper, we propose to

use the distributed constraint optimization (DCOP) [12] to model the local interaction structure of agents and decentralized execution [12]. However, one of the most significant barriers to decentralized approaches is the mismatch between an individual action and the global reward [13], [14]. Thus, the key challenge in decentralized approaches is assessing how a single agent's action contributes to the global payoff, which is known as the structural credit assignment (SCA) problem. SCA can be classified into two categories: *implicit SCA* and *explicit SCA*. Implicit SCA learns the relationship between individual and global state-action values using a decomposition neural network [15], [16]. Implicit SCA requires an infeasible amount of experience and is only applied to reinforcement learning (MARL) tasks. The difference reward is a widely used explicit SCA that rewards each agent by the difference between the global reward and counterfactual global reward without this agent [17], [18], [19], [20]. However, the conventional explicit SCA does not provide fair and efficient contributions for agents [21].

In this study, we design a new explicit SCA using the Shapley value (SV-SCA), a concept that originates from cooperative game theory, and is theoretically guaranteed to distribute the global payoff efficiently and fairly to agents [22]. A major drawback of the Shapley value is that it cannot be computed in polynomial time [23]. Several approximation heuristics have been developed to overcome the problem of computational hardness in determining the exact Shapley value [21], [24], [25]; however, they also violate the efficiency and fairness properties of the SCA [26]. In DCOP, each agent's contribution to the overall system is only relevant to the local agents, and the computational complexity of the Shapley value can be significantly reduced by exploiting such local interactions.

Multiagent planning (MAP) and multiagent RL (MARL) are two popular MMDP methodologies that complement each other and each has individual advantages [27]. Combining the Shapley value with DCOP not only inherits the desirable properties of the traditional Shapley value, but also yields a generalized SCA that can be applied to both MAP and MARL tasks. First, guided by SV-SCA, we propose an efficient and scalable coordinated MCTS method for online MAP. The technical contribution of the coordinated MCTS is using SV-SCA to backpropagate the simulated the future global reward to individual agents at the root state. Using DCOP, we can calculate the Shapley value efficiently such that the scalability of the coordinated MCTS is preserved. Theoretical analyses show that the proposed coordinated MCTS can guarantee the expected value of the global joint action Second, we propose a coordinated MARL, in which the global state-action value Q_{tot} can be approximated by the sum of local Q-values modeled by DCOP. The local Q-value is determined by SV-SCA in a fully decentralized manner, and is aligned with the global Q_{tot}-value. Theoretical analyses show that the proposed coordinated MARL is monotonic, which provides a sufficient condition for convergence to the optimal global joint policy.

The first contribution of this paper is a general SV-SCA framework by combining DCOP and Shapley value. The proposed SV-SCA allows decentralized execution and provides an efficient evaluation of individual actions. Based on SV-SCA, the second contribution is a decentralized coordinated MCTS for MAP. Although DCOP is state-dependent, we can use Shapley value to decompose the future global rewards to each individual agents, and propagate the future global rewards to individual agents at the root state. The third contribution is a decentralized coordinated MARL. At each state, we can use SV-SVA to decompose the global reward into individual rewards, and allow each agent to learn its own policy based on its individual critic. Finally, to validate the proposed MAP and MARL methods based on SV-SCA, extensive experiments are conducted in a series of sequential coordination domains. Experimental results show that SV-SCA based coordinated MCTS and MARL outperform existing benchmarks for multiagent MCTS and MARL in terms of solution quality and scalability.

The remainder of this paper is organized as follows. In Section II, we review the related literature on MAP and MARL. In Section III, we formulate the sequential coordination problem as an MMDP. In Section IV, we combine the Shapley value and DCOP, and propose a new SCA for individual action evaluation. Guided by SV-SCA, we propose a coordinated MCTS method for online MAP and a coordinated and fully decentralized method for MARL in Sections V and VI, respectively. In Section VII, we describe a series of experiments conducted to validate the proposed coordinated MCTS and MARL methods based on SV-SCA. Section VIII discusses the limitations of the proposed SV-SCA, coordinated MCTS and MARL methods. Finally, we conclude the paper in Section IX.

II. Related Work

Concerning works on MAP, we mainly focus on researches based on MCTS, one of the most powerful online planning methods. Concerning works on MARL, we mainly focus on structural credit assignment mechanisms used to decompose global state-action values into individual values during training.

A. MCTS-Based Online MAP

MCTS is a well-known online planning strategy for constructing real-time actions for sequential decision-making problems [34] such as MDP [35], partially observable MDP [36], and continuous control tasks [37]. As the number of global joint actions grows exponentially, directly applying MCTS to MMDP is difficult for balancing the exploration and exploitation. To reduce the search complexity, *single-agent MCTS* allows only one particular agent to search for the policy, whereas all other agents follow a fixed predetermined policy [38]. Decentralized MCTS (*Dec-MCTS*) enables all agents to search for their own policies while considering the behaviors of other agents [39]. In Dec-MCTS, the behaviors of other agents can be modeled using greedy heuristics [40], [41] or learned from prior experience [42], [43]. To improve the modeling accuracy, agents can communicate and share their policies with other agents [44], [45]. However, these independent MCTS methods can result in suboptimal solutions

TABLE I
SUMMARY OF THE SCA-BASED MARL METHODS ON GLOBAL Q-VALUE AND Q-VALUE DECOMPOSITION

Category \ Property		Global Q-value Q_{tot}	Q-value Decomposition
Implicit SCA	VDN-based [15], [16], [28], [29], [30]	Neural Network	Neural Network
Explicit SCA	COMA [19], [31] and SHAQ [24], [21], [25], [32], [33]	Neural Network	Difference and Shapley-value
	Our SV-SCA	None	Shapley-value

in complex multi-agent problems, where reasoning regarding the effect of joint actions is necessary [46]. The proposed *coordinated MCTS* attempts to explore and evaluate joint actions to improve the quality of the solutions.

In several real MASs, agents interact only with a subset of local agents, and this local interaction can be modeled using a coordination graph (CG) [12]. More recently, *FV-MCTS* in conjunction with CGs explored local joint actions according to upper confidence bound statistics [46], [47]. Because CGs are dynamic and state-dependent, the main challenge of the FV-MCTS is the exploration of local joint actions while maximizing the cumulative long-term global payoff. Existing FV-MCTS directly uses the global payoff (which can be computed exactly by variable elimination [47] or approximately by max-plus [46]) to reward each agent. This individual reward is then used to evaluate the local joint action value. However, because each agent's reward depends on the actions of all other agents, global payoff-based credit assignment suffers from low-payoff noncooperative solutions in complex environments with a number of exploratory agents [18]. To address these limitations, our goal is to design a SV-SCA-guided *coordinated MCTS*, where SV-SCA is used to promote coordination and MCTS is used to select the most promising coordinated joint action. Because SV-SCA can be computed efficiently, the proposed coordinated MCTS can preserve the anytime property.

B. SCA for MARL

The centralized MARL learns a global state-action value for all agents, might converge to suboptimal solutions as these *lazy* agents hinder the active agent's policy. Therefore, state-of-the-art MARL methods have widely adopted the paradigm of centralized training for decentralized execution (CTDE), in which agents are trained offline using centralized information but executed in a decentralized manner online [48], [49].

SCA is one of the key challenges in CTDE; specifically, decomposing the global payoffs/values to each agent such that they can be more coordinated to maximize the system performance is challenging [20], [50], [51]. Prior SCAs for MARL can be divided into two categories: *implicit* and *explicit*. Table I gives a summary of the SCA-based MARL methods.

1) Implicit SCA Methods: In implicit methods, value decomposition networks (VDNs) [15] represent the global state-action value Q_{tot} as a sum of individual value functions Q_i, QMIX [16] represents Q_{tot} as a linear weighted sum of Q_i and FACMAC [52] relaxes the constraints on factoring the critic. QTRAN [29] presents generalized value factorization that guarantees the property of individual-global-max (IGM) between Q_{tot} and Q_i. To address the computational complexity of decomposing the global Q_{tot}, QPLEX introduces the dueling structure $Q_{tot} = V_{tot} + A_{tot}$ for representing both joint and individual action-value functions and then reformalizes the IGM principle as an advantage-based IGM [30]. Deep coordination graphs factorize the global Q_{tot} according to a CG into payoffs between pairs of agents in a linear [53] and non-linear manner [54]. Li et al. [55] use the self-attention mechanism to learn a fully connected CG, and Wang et al. [56] construct sparse CGs using the variance of payoff functions. However, these implicit SCA methods either approximate Q_{tot} as the weighted sum of local Q-values (e.g., VDN [15], QMIX [16] and DIGC [53]), or models the non-linear relationships (e.g., QTRAN [29], and QPLEX [30]) based on neural networks. These implicit SCA methods require a large number of simulations and are unable to provide theoretical guarantees of convergence. In contrast to VDN and QMIX, this paper proposes the Shapley value technique to decompose the true global payoffs to individual rewards, which satisfies the properties of additivity, monotonicity, and fairness. In contrast to QTRAN and QPLEX, this paper does not compute the global Q_{tot}-value, and has the advantage of scalability.

2) Explicit SCA Methods: Agogino and Tumer [17], [18] first proposed the difference reward for SCA, in which each agent is rewarded by the difference between the global payoff and counterfactual global payoff without this agent. Inspired by the difference evaluation function, Nguyen et al. [20], Foerster et al. [19] and Liu et al. [31] proposed the counterfactual critic for each agent. To prompt individual exploration as well as reduce variance, Du et al., [57] integrates the intrinsic reward for each agent and Xu et al. [58] design a local critic to train each agent's policy. However, the SCA and the counterfactual critic based on difference reward needs to learn the global state-action value Q_{tot}, and thus cannot be applied to problems with a large number of agents. Recently, leveraging concepts of cooperative game theory, the Shapley value has been proposed as an alternative critic Q_i^{ϕ} for each agent, where Q_i^{ϕ} is computed as the average contribution of the marginal state-action value $Q_{\mathcal{C}}$ to any coalition $\mathcal{C}$ [21], [24], [25], [32], [33].

However, existing difference and Shapley value-based explicit MARL have two challenges: 1) Shapley value-based critic still requires to learn global state-action value Q_{tot} and 2) computing the exact Shapley value is intractable, and these sampling-based approximations [24], [25], [32] violate the efficiency and fairness of the Shapley value. By contrast, based on the DCOP model, the proposed SV-SCA can exploit the local interaction structure and compute the exact Shapley value

efficiently. Additionally, by combining the Shapley value with the DCOP, we can decompose the global payoff into individual rewards in a completely decentralized manner that follows a more scalable decentralized training execution framework [59].

III. Problem Description and Algorithm Framework

A. MMDP Model

Formally, an MMDP can be defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{N} = \{1, 2, \cdots, n\}$ is the set of n agents and $\mathcal{S}$ is a finite set of global states s of the environment. In the MMDP, each agent observes a global state. $\mathcal{A} = A_1 \times \cdots \times A_n$ denotes the set of joint actions $\vec{a} = \langle a_1, \cdots, a_n \rangle$. $P(s, \vec{a}, s') \in \mathcal{P}$ is the transition probability of ending at state s' given that the joint action $\vec{a}$ is applied at s. $R(s, \vec{a}, s') \in \mathcal{R}$ is the immediate global reward for taking joint action $\vec{a}$ at state s and ending at state s'.

In an MMDP, a joint policy $\pi : \mathcal{S} \to \mathcal{A}$ maps the global state to the global joint action $\vec{a}$. This joint policy $\pi = \times_{1 \le i \le n} \pi_i$ is equivalent to a tuple of individual policies $\pi_i : S \to A_i$. The agents interact with the environment sequentially for T periods. A finite sequence $\rho_\pi = \langle s_0, s_1, \cdots, s_T \rangle$ of states generated by the policy π is called a trajectory. For any joint policy π in an MMDP, the expected discounted cumulative global return at state s is $V^\pi(s) = \mathbb{E}_{\rho_\pi \sim \pi}\left[\sum_{k=0}^{T-1} \gamma^k R(s_k, \vec{a}_k, s_{k+1}) | s_0 = s\right]$, where $\mathbb{E}$ denotes the expected value by following π, and $\gamma \in [0, 1]$ is the discount factor. The objective of solving an MMDP is to determine a joint policy π that can generate a trajectory ρ_π to maximize the expected cumulative global return $V^\pi(s_0)$ at the starting state s_0.

B. The Framework SV-SCA for MAP and MARL

MAP and MARL are two popular approximations for MMDP that complement each other. Fig. 1 illustrates the flow of SV-SCA based online MAP and MARL, comprising the following three parts:

- **Part 1: SV-SCA for Individual Action Evaluation**. In this part, we consider a multiagent coordination problem with a single state, which is the foundation of the sequential MMDP problems. The aim of this part is to search for the optimal joint-action that can maximize the global payoff, and then decompose the optimal global payoff for each agent. The main contribution of this part is to propose an efficient algorithm for computing the exact Shapley value, which can provide an efficient and fair evaluation of individual actions.
- **Part 2: SV-SCA Guided Coordinated MCTS for Online MAP**. In this part, MCTS is used to find the most promising global joint action that can maximize the cumulative long-term global return. The main contribution of this part is that SV-SCA is used to decompose the state-independent global payoff to reward agents, and the reward of each agent then is backpropagated for evaluating the local joint action.
- **Part 3: SV-SCA Guided Coordinated MARL**. Coordinated MARL allows agents to learn their own state-action value in a distributed manner, but coordinates distributed learning by exploiting local interactions. The main contribution of this part is that SV-SCA is used to align the local Q-value of the local joint-action with the global Q_{tot}-value.

IV. SV-SCA for Individual Action Evaluation

In this section, we first review the multiagent coordination model, which is used to describe local interactions among agents. Next, we describe the SV-SCA used to evaluate individual actions, and demonstrate the use of the multiagent coordination model to compute the Shapley value exactly.

DCOP. Several multiagent coordination problems demonstrate the *locality of interaction*; that is, an agent's action only affects the actions of a subset of agents that interact locally. DCOP has emerged as a key formalism for settings in which primary interactions occur between local subsets of agents [12]. Formally, a DCOP can be defined as a tuple $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ such that: $\mathcal{N} = \{1, 2, \cdots, n\}$ is the set of agents, $\mathcal{A} = \{a_1, a_2, \cdots, a_n\}$ is the set of variables, each a_i is controlled by an agent i, and takes the value from the finite discrete domain $D_i \in \mathcal{D}$[1], and $\mathcal{U} = \{u_1, u_2, \cdots, u_m\}$ is a set of local utilities, each u_j is defined as a mapping from the assignments of the involved k variables $\vec{a}_{u_j} = (a_{j_1}, \cdots, a_{j_k})$ to a positive real number,[2] $u_j : D_{j_1} \times D_{j_2} \times \cdots \times D_{j_k} \to \mathbb{R}^+$. Let $\mathcal{N}_{u_j}$ denote the set of agents involved in utility u_j. Without loss of generality, let $\emptyset$ denote the null operation action of agents and $\forall u_j$, $u_j(\emptyset) = 0$; that is, there will be no utility if all agents do nothing. A solution to a DCOP is to find a global joint action $\vec{a}^*$ that maximizes the global payoff $u(\mathcal{N}) = \sum_{u_j \in \mathcal{U}} u_j(\vec{a}_{u_j})$, which is the sum of all local utility functions, that is, $\vec{a}^* = \arg\max_{\vec{a}} u(\mathcal{N})$.

Given the local interaction, an agent i's action affects only the utility functions that involve i. Next, we provide a formal definition of *local interacting agents* and *irrelevant agents*, which are useful for computing the Shapley value.

Definition 1 (Local Interacting Agents and Irrelevant Agents) Given a DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$, an agent k is defined as the local interacting agent of agent i if there exists at least one utility function that involves both i and k, that is, the set of local interacting agents of i, $\mathcal{N}_i^{loc} = \{k | \exists u_j \in \mathcal{U} : i \in \mathcal{N}_{u_j} \& k \in \mathcal{N}_{u_j}\}$. On the other hand, an agent k is defined as the irrelevant agent of agent i if there is no utility function that involves both i and k, that is, the set of irrelevant agents of i, $\mathcal{N}_i^{irr} = \mathcal{N} \setminus \mathcal{N}_i^{loc}$.

Max-sum for Optimizing DCOP. Given a DCOP, finding a globally optimal solution is NP-hard [60]. A Max-sum algorithm is proposed to optimize the global joint action $\vec{a}^*$ in a fully decentralized manner [61]. The pseudocode for the standard Max-sum is presented in Algorithm 1. The Max-sum first transforms the DCOP into a factor graph: a bipartite undirected graph that contains a variable node for each agent i, a function node for each utility function u_j, and an edge-connecting variable node i with function node u_j if and only if i

[1]In multiagent decision making domains, D_i represents the set of actions.
[2]In multiagent decision making domains, the k variables represent the actions of k agents.

Algorithm 1 Max-sum(0)

Input : The DOCP $\langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$.
Output: Joint action $\vec{a}^*$.

```
for T iterations do
    for variable node i do
        produce message E_{i→u_j} using messages from Neg_i \ {u_j}.
    for funtion node u_j do
        produce message F_{u_j→i} using messages from Neg_{u_j} \ {i}.
Compute the joint action a⃗* using Eq.(3).
```

is involved in u_j. The variable and function nodes can send and read messages and perform computations. In a factor graph, let Neg_i (resp. Neg_{u_j}) denote the set of neighbor function nodes (variable nodes) of variable node i (function node u_j). The operations of the variable and function nodes are similar, except for the content of the messages to be sent. A message sent from a variable node i to a function node u_j at iteration t, includes for each value $a_i \in D_i$ the sum of utilities for this value it received from all function neighbors except for u_j at iteration $t-1$. Formally, at the tth iteration, the message sent from the variable node i to the function node u_j includes each action $a_i \in D_i$:

$$E_{i\to u_j}(a_i) = \sum_{u_{j'}\in Neg_i, u_{j'}\neq u_j} F_{u_{j'}\to i}(a_i) - \alpha, \tag{1}$$

where $F_{u_{j'}\to i}(a_i)$ is the utility of the value a_i included in the messages received from function node $u_{j'}$ at iteration $t-1$, and α is a constant that prevents utilities carried by messages from growing arbitrarily. A common choice of α is the average of all utilities included in the message; that is, $\alpha = \sum_{a_i\in D_i}\sum_{u_{j'}\in Neg_i, u_{j'}\neq u_j} F_{u_{j'}\to i}(a_i)/|D_i|$.

A message sent from function node u_j to variable node i at iteration t includes for each possible value $a_i \in D_i$ the maximal utility of any combination of actions to the variables involved in u_j except for i. Formally, the message sent from function u_j to variable i includes the following for each action $a_i \in D_i$:

$$F_{u_j\to i}(a_i) = \max_{\vec{a}_{u_j}\setminus a_i} \left(u_j(\vec{a}_{u_j}) + \sum_{k\in Neg_{u_j}, k\neq i} E_{k\to u_j}(a_k) \right), \tag{2}$$

where $u_j(\vec{a}_{u_j})$ is the local utility function involving local joint action $\vec{a}_{u_j}$.

When a variable node makes decisions, it accumulates all utilities it receives and selects an action to maximize the sum of the utilities. Formally, each variable node i selects an action using

$$a_i^* = \arg\max_{a_i} \sum_{u_j\in Neg_i} F_{u_j\to i}(a_i). \tag{3}$$

The Max-sum algorithm can converge to the optimum for the DCOP with an acyclic factor graph, and can also provide a desirable solution in cycle factor graphs [61].

A. The SV-SCA Mechanism

Given a DCOP and the optimal joint-action $\vec{a}^*$, the SCA problem aims to evaluate how the individual action a_i^* of agent i contributes to the global payoff. In this section, we describe the core idea of our SV-SCA and demonstrate the exploitation of the structure of the DCOP in computing the Shapley value efficiently.

1) Marginal Contribution: Given a DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$, let $\mathcal{C} \subseteq \mathcal{N}$ denote a coalition, that is, a subset of agents. Given the global joint action $\vec{a} = \{a_1, \cdots, a_n\}$ returned by Max-sum, let $u(\mathcal{C}) = \sum_{u_j} u_j(a_{j_1}^{\mathcal{C}}, \cdots, a_{j_k}^{\mathcal{C}})$ denote the total utility achieved by the coalition $\mathcal{C}$, in which the agent i takes the action

$$a_i^{\mathcal{C}} = \begin{cases} a_i & \text{if the agent participates in } \mathcal{C}, \\ \emptyset & \text{otherwise.} \end{cases} \tag{4}$$

The marginal contribution $\triangle_i^{\vec{a}}$ of agent i to coalition $\mathcal{C} \subseteq \mathcal{N}\setminus\{i\}$ is the increase in the utility of $\mathcal{C}$ as a result of i joining it and taking action a_i, that is,

$$\triangle_i^{\vec{a}}(\mathcal{C}) = u(\mathcal{C}\cup\{i\}) - u(\mathcal{C}). \tag{5}$$

2) Shapley Value: The Shapley value of agent i taking action a_i, $\phi_i^{\vec{a}}$ is the average marginal contribution to all possible coalitions.

$$\phi_i^{\vec{a}} = \sum_{\mathcal{C}\subseteq\mathcal{N}\setminus\{i\}} \frac{|\mathcal{C}|!(n-|\mathcal{C}|-1)!}{n!} \triangle_i^{\vec{a}}(\mathcal{C}). \tag{6}$$

The Shapley value can be interpreted as follows: all agents are arranged in some order, all orderings are equally likely, and $\phi_i^{\vec{a}}$ is the expected marginal contribution over all orders of agent i to the set of preceding agents.

The SV-SCA mechanism assigns credit to agent i using the Shapley value $\phi_i^{\vec{a}}$. This type of credit assignment not only provides a fair division of the utilities of coordination among agents, but also satisfies the *additivity* property [23].

Lemma 1: [Additivity [23]] Give a DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$, let $\vec{a}$ denote any global joint-action, then the global payoff generated by the grand coalition $\mathcal{N}$ is equal to the sum of the Shapley value-based credits of all agents, $\sum_i \phi_i^{\vec{a}} = u(\mathcal{N})$,

3) Computing the Shapley Value Efficiently: Although the Shapley value provides a fair SCA, its main drawback is its computational complexity. Fortunately, in the DCOP, agent i's action affects only the utility of the coalition involving the locally interacting agents of i. This observation indicates that i's marginal contributions to all coalitions involving the same subset of locally interacting agents of i is equal. In the following, we consider this insight and compute the exact Shapley value efficiently.

Theorem 1: Given the agent i, let $\mathcal{N}_i^{loc}$ denote the subset of the locally interacting agents (defined in Definition 1) and $n_i^{loc} = |\mathcal{N}_i^{loc}|$, and $\mathcal{N}_i^{irr} = \mathcal{N}\setminus\mathcal{N}_i^{loc}$ denote the subset of irrelevant agents that do not interact with the agent i and $n_i^{irr} = |\mathcal{N}_i^{irr}|$. The Shapley value of i, $\phi_i^{\vec{a}}$ can then be rewritten as

$$\phi_i^{\vec{a}} = \frac{1}{n!} \sum_{\mathcal{C}\subseteq\mathcal{N}_i^{loc}} H(|\mathcal{C}|, n_i^{irr}) \triangle_i^{\vec{a}}(\mathcal{C}), \tag{7}$$

where the operator $H(|\mathcal{C}|, n_i^{irr})$ depends only on the *number* of locally interacting agents and the *number* of irrelevant agents.

$$H(|\mathcal{C}|, n_i^{irr}) = \sum_{k=0}^{n_i^{irr}} \binom{n_i^{irr}}{k} (|\mathcal{C}| + k)!(n - |\mathcal{C}| - k - 1)!$$

Proof: For each coalition $\mathcal{C}$, dividing it into disjoint coalitions of locally interacting agents $\mathcal{C}_1 \subseteq \mathcal{N}_i^{loc}$ and irrelevant agents $\mathcal{C}_2 \subseteq \mathcal{N}_i^{irr}$ of agent i, we have

$$\begin{aligned}
\phi_i^{\vec{a}} &= \frac{1}{n!}\sum_{\mathcal{C}\subseteq\mathcal{N}\setminus\{i\}} |\mathcal{C}|!(n - |\mathcal{C}| - 1)!\triangle_i^{\vec{a}}(\mathcal{C}) \\
&= \frac{1}{n!}\sum_{\mathcal{C}_1\subseteq\mathcal{N}_i^{loc}} \sum_{\mathcal{C}_2\subseteq\mathcal{N}_i^{irr}} |\mathcal{C}_1\cup\mathcal{C}_2|!(n-|\mathcal{C}_1 \cup \mathcal{C}_2|-1)!\triangle_i^{\vec{a}} \\
&\quad \times (\mathcal{C}_1 \cup \mathcal{C}_2) \\
&= \frac{1}{n!}\sum_{\mathcal{C}_1\subseteq\mathcal{N}_i^{loc}} \sum_{\mathcal{C}_2\subseteq\mathcal{N}_i^{irr}} |\mathcal{C}_1 \cup \mathcal{C}_2|!(n-|\mathcal{C}_1 \cup \mathcal{C}_2|-1)!\triangle_i^{\vec{a}}(\mathcal{C}_1) \\
&= \frac{1}{n!}\sum_{\mathcal{C}_1\subseteq\mathcal{N}_i^{loc}} \sum_{\mathcal{C}_2\subseteq\mathcal{N}_i^{irr}} (|\mathcal{C}_1|+|\mathcal{C}_2|)!(n-|\mathcal{C}_1|-|\mathcal{C}_2|-1)! \\
&\quad \times \triangle_i^{\vec{a}}(\mathcal{C}_1) \\
&= \frac{1}{n!}\sum_{\mathcal{C}_1\subseteq\mathcal{N}_i^{loc}} \sum_{k=0}^{n_i^{irr}} \binom{n_i^{irr}}{k} (|\mathcal{C}_1|+k)!(n-|\mathcal{C}_1|-k-1)!\triangle_i^{\vec{a}}(\mathcal{C}_1) \\
&= \frac{1}{n!}\sum_{\mathcal{C}_1\subseteq\mathcal{N}_i^{loc}} H(|\mathcal{C}_1|, n_i^{irr})\triangle_i^{\vec{a}}(\mathcal{C}_1). \qquad (8)
\end{aligned}$$

In Eq. (8), given a local interacting coalition $\mathcal{C}_1 \subseteq \mathcal{N}_i^{loc}$, for any irrelevant coalition $\mathcal{C}_2 \subseteq \mathcal{N}_i^{irr}$, the agent i has the same marginal contribution to the joint coalition $\mathcal{C}_1 \cup \mathcal{C}_2$, that is, $\forall \mathcal{C}_2, \mathcal{C}_2' \subseteq \mathcal{N}_i^{irr}$, $\triangle_i^{\vec{a}}(\mathcal{C}_1 \cup \mathcal{C}_2) = \triangle_i^{\vec{a}}(\mathcal{C}_1 \cup \mathcal{C}_2')$. □

It should be noted that the operator $H(|\mathcal{C}|, n_i^{irr})$ depends only on the *number* of local coalitions (that is, $2^{n_i^{loc}}$) and the *number* of irrelevant agents n_i^{irr}, rather than the agents' actions. Thus, we can precompute $H(x, y)$ $(1 \le y \le n, 0 \le x \le n - y)$ to mitigate the online computation load.

In conclusion, Eq. (1) has two main advantages: 1) the Shapley value can be computed efficiently by reducing the number of possible coalitions from 2^n to $\sum_{i\in\mathcal{N}} 2^{n_i^{loc}}$, and 2) each agent can compute its Shapley value by accessing only the local joint-action in a fully decentralized manner without the need for knowledge of the global joint-action.

V. SV-SCA Guided Coordinated MCTS for Online MAP

In this section, we combine SV-SCA with MCTS and propose a coordinated MCTS, which is an online MAP method for MMDPs. Searching for the global joint-action is time consuming, and the coordination structure of DCOP changes with the state; therefore, the key to implementing a coordinated MCTS is individual action selection and global payoff backpropagation. The overview framework of coordinated MCTS is shown in Fig. 2.

A. Individual Action Selection

Given the current state s, we first formulate the current DCOP $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ at s. On the one hand, agents need to coordinate their joint-action $\vec{a}$ by a Max-sum to optimize the immediate global payoff. However, it is also necessary to explore promising joint actions that will benefit long-term cumulative global payoffs. As previously stated, it is not feasible to directly explore the global joint-action. Using the message-passing procedure in DCOP, we can explore the individual action a_i of agent i using Eq. (3) at the end of the process. Specifically, we track the corresponding frequency statistics for every action a_i of agent i. A natural exploration of an individual action is to add an upper confidence bound (UCB) bonus when selecting an action:

$$a_i^* = \arg\max_{a_i} \left[\sum_{u_j \in Neg_i} F_{u_j \to i}(a_i) + c\sqrt{\frac{\ln N(s)}{N_i(s, a_i)}} \right], \qquad (9)$$

where c is the exploration parameter, $N_i(s, a_i)$ is the number of visits to the individual action a_i of agent i in state s, and $N(s)$ is the total number of visits to state s.

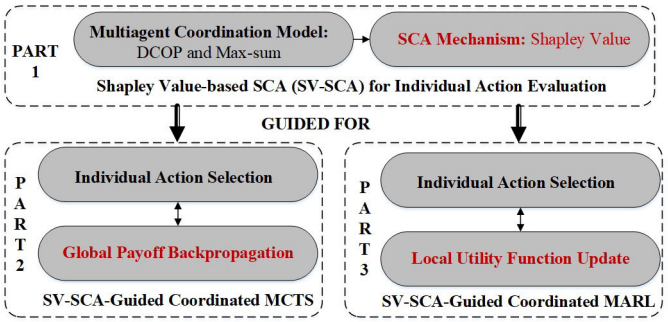


Fig. 1. The framework of SV-SCA-guided coordinated MCTS for online MAP and coordinated MARL. The elements highlighted in red are the contributions of this study.

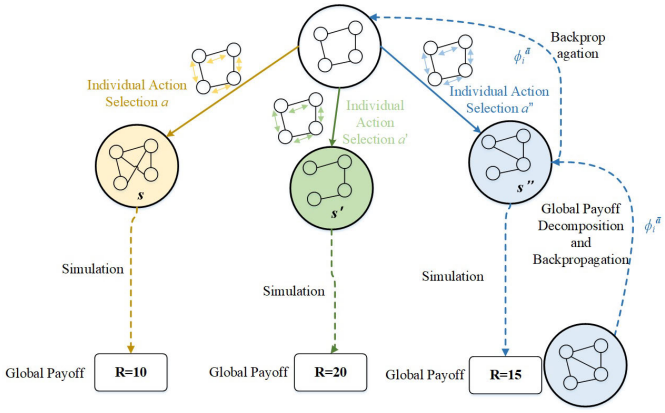


Fig. 2. Overview of coordinated MCTS.

B. Global Payoff Backpropagation

In a standard single-agent MCTS, the value of all child nodes in the root state s is updated by backpropagating the payoff from the leaf node to the root node s. However, in an MMDP, it is not possible to explore the global joint-action, and it is not possible to directly backpropagate the global payoff to update global joint-action statistics. To backpropagate the global payoff, we use SV-SCA to decompose the global payoff into individual rewards and backpropagate the individual rewards to update the statistics of individual actions as well as local utility functions.

C. Updating Individual Action and Local Utility Function

Given the current DCOP $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ at state s, let $\vec{a} = (a_1, \cdots, a_n)$ denote the global joint-action, where a_i is selected using Eq. (9), and $R(s, \vec{a}, s')$ denotes the immediate global payoff achieved by taking $\vec{a}$ and ending in the state s'. We first use SV-SCA to determine the immediate individual reward $R_i(s, a_i, s')$ of agent i, that is, $\forall i$:

$$R_i(s, a_i, s') = \phi_i^{\vec{a}}, \tag{10}$$

where $\phi_i^{\vec{a}}$ denotes the Shapley values defined in Eq. (1). In a coordinated MCTS, the key is to model each of the local utility functions $u_j \in \mathcal{U}$ at state s, which will be used for the coordinated action selection in Eq. (3). Let $q_i(s, a_i)$ denote the discounted cumulative individual reward of i by taking action a_i in state s. The discounted cumulative rewards of each local utility function u_j, $q_{u_j}(s, \vec{a}_{u_j})$ can be defined as the sum of the weighted cumulative individual rewards of agent $i \in Neg_{u_j}(s)$:

$$q_{u_j}(s, \vec{a}_{u_j}) = \sum_{i \in Neg_{u_j}(s)} \frac{q_i(s, a_i)}{|Neg_i(s)|}, \tag{11}$$

where $Neg_{u_j}(s)$ denotes the neighbor agents of u_j and $Neg_i(s)$ denotes the neighbor utility function of agent i at the current DCOP $\mathcal{G}(s)$. This method of updating the local utility function preserves the efficiency of the Shapley value, shown as follows:

Lemma 2: Given a DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$, let $\vec{a}$ denote a global joint-action, $u(\mathcal{N})$ denote the global payoff achieved by $\vec{a}$, and $\phi_i^{\vec{a}}$ denote the Shapley value of agent i. For each local utility function $u_j \in \mathcal{N}$, let $q_{u_j}(\vec{a}_{u_j}) = \sum_{i \in Neg_{u_j}} \frac{\phi_i^{\vec{a}}}{|Neg_i|}$ denote the reshaped value. By summing the reshaped values of all utility functions, we also guarantee the efficiency property; that is, $u(\mathcal{N}) = \sum_{u_j \in \mathcal{U}} q_{u_j}(\vec{a}_{u_j})$.

Finally, we define $\overline{u}_j(s, \vec{a}_{u_j})$, the mean of the local utility function u_j obtained when the local joint-action $\vec{a}_{u_j}$ is selected at state s.

$$\overline{u}_j(s, \vec{a}_{u_j}) = \overline{u}_j(s, \vec{a}_{u_j}) + \frac{q_{u_j}(s, \vec{a}_{u_j}) - \overline{u}_j(s, \vec{a}_{u_j})}{N_{u_j}(s, \vec{a}_{u_j})}, \tag{12}$$

where $N_{u_j}(s, \vec{a}_{u_j})$ denotes the visit frequency of the local joint-action $\vec{a}_{u_j}$. The average local utility $\overline{u}_j(s, \vec{a}_{u_j})$ represents the expected cumulative long-term utility function used in the message-passing procedure (Eqs. (1– 2)) for individual action selection.

D. The Algorithm

The SV-SCA-based coordinated MCTS algorithm is formally shown in Algorithm 2, which cycles the INITIALIZESTATE, SIMULATE, MAX-SUM, and UPDATESTATE functions.

- INITIALIZESTATE(s) (Steps 13-19): For a new state s, we first build the DCOP model $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ (Step 15). We then initialize the frequency statistics of each single action $N_i(s, a_i)$ and each local joint-action $N_{u_j}(s, \vec{a}_{u_j})$, and the mean value of the local utility function u_j for each local joint-action $\vec{a}_{u_j} \overline{u}_j(s, \vec{a}_{u_j})$ (Steps 16-19).
- SIMULATE(s, *depth*) (Steps 4-12): Starting from the current state s, the SIMULATE function incrementally grows a search tree of simulating and evaluating local utilities until *depth* or time limit is reached. At Step 8, MAX-SUM(c) is used to select the global joint-action $\vec{a}$ and transit to state s'. Given the global joint-action $\vec{a}$, we use the SV-SCA to determine the individual reward for each agent i, that is, $R_i(s, a_i, s') = \phi_i^{\vec{a}}$ (Step 10). At Step 11, by further simulation, the cumulative state action reward $q_i(s, a_i)$ of the agent i is computed by summing the discounted future individual reward $\gamma \cdot$ SIMULATE(s', depth $-$ 1)[q_i] of i.
- MAX-SUM(c) (Steps 20-22): Once the message passing procedure (that is, Steps 1-5 in Max-sum(0)) terminates,[3] each agent selects its own action by Eq. (9), where the parameter c yields a trade-off between exploration and exploitation.
- UPDATESTATE(s,$\vec{a}$,$\overline{u}$) (Steps 23-29): At the state s, given the global joint-action $\vec{a}$, for each u_j, the cumulative reward $q_{u_j}(s, \vec{a}_{u_j})$ of the local joint-action $\vec{a}_{u_j}$ is computed by summing the weighted individual rewards of neighboring agents $i \in Neg_{u_j}$ (that is, Eq.(11)). Finally, the mean statistics of u_j with respect to the local joint-action $\vec{a}_{u_j}$, $\overline{u}_j(s, \vec{a}_{u_j})$ is then updated by Eq. (12).

E. Theoretical Analysis

In this section, we first analyze the computational complexity of coordinated MCTS.

Lemma 3: Assume that there are n agents, each agent has d actions and k neighbor agents, the computation complexity of Algorithm 2 is $O(d^2n^2 + Tk^2d^k + n2^k)$, where T is the number of iterations in Max-sum (i.e., Algorithm 1).

Proof: Algorithm 2 is implemented by simulations (Steps 1-2). Each simulation consists of three functions: INITIALIZESTATE (Steps 13-19), MAX-SUM (Steps 20-22), UPDATESTATE (Steps 23-29).

INITIALIZESTATE (Steps 13-19): at each state s, we need to take $O(kn)$ operations to formulate DCOP. For each agent i, we need to initialize its visit frequency $N(s, a_i)$ for each action a_i, which will take $O(dn)$ operations. For each utility u_j, and each joint action $\vec{a}_{u_j}$, we need to initialize its visit frequency $N_{u_j}(s, \vec{a}_{u_j})$, which will take $O(d^2n^2)$ operations. In total, there are $O(kn + dn + d^2n^2) = O(d^2n^2)$ operations in INITIALIZESTATE.

MAX-SUM (Steps 20-22): Max-Sum is implemented in iterations. At each iteration (Steps 1-5 in Algorithm 1), each agent i computes the coordination cost $E_{i \to u_j}$ for each utility function u_j and each action a_i. Since there are k neighbors and d actions for each agent, there are $O(kd)$ computations for each agent to compute $E_{i \to u_j}$. On the other hand, each utility u_j computes the maximal utility function $F_{u_j \to i}$ (i.e., Eq. (2)) to each neighbor agent i with respect to each action a_i, which will take $O(kd^k)$ operations. Since there are k neighbor agents, there are $O(k^2d^k)$ computations for each utility function u_j. MAX-SUM terminates at most T iterations, the total computation cost of MAX-SUM is $O(T(kd + k^2d^k)) = O(Tk^2d^k)$.

[3]It should be noted that Max-sum is an anytime algorithm that can be terminated at any time (for time limit) and return the current best solution.

Algorithm 2 SV-SCA Based Coordinated MCTS

Input : time limit, *depth* of tree search, the exploration parameter c, state s, discounted factor γ.

Output: Global Joint Action $\vec{a}^*$.

1 **while** *time limit not reached* **do**
2 SIMULATE(s,*depth*);
3 $\vec{a}^* \leftarrow$ Max-sum(0) by Algorithm 1;
4 **Function** SIMULATE (s,*depth*) **:**
5 **if** *depth=0* **then**
6 return 0;
7 INITIALIZESTATE(s);
8 $\vec{a} \leftarrow$ MAX-SUM(c), $s' \sim P(s, \vec{a}, s')$;
9 **for** *each agent* $i \in \mathcal{N}$ **do**
10 $R_i(s, a_i, s') = \phi_i^{\vec{a}}$ via Eq. (1);
11 $q_i(s, a_i) \leftarrow$ $R_i(s, a_i, s') + \gamma \cdot$ SIMULATE$(s', depth-1)[q_i]$;
12 UPDATESTATE(s,$\vec{a}$,$\overline{u}$);
13 **Function** INITIALIZESTATE (s) **:**
14 **if** s *is a new state* **then**
15 We formulate $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$.
16 **for** *each agent* $i \in \mathcal{N}$ & $a_i \in D_i$ **do**
17 We initialize $N_i(s, a_i) = 0$;
18 **for** *each utility function* $u_j \in \mathcal{U}$ & $\vec{a}_{u_j} \in D_{u_j}$ **do**
19 Initialize $\overline{u}_j(s, \vec{a}_{u_j}) = 0$, and $N_{u_j}(s, \vec{a}_{u_j}) = 0$;
20 **Function** MAX-SUM (c) **:**
21 Steps 1-5 of the Max-sum(0) using Algorithm 1;
22 The global joint-action $\vec{a}^*$ is computed using Eq.(9);
23 **Function** UPDATESTATE (s,$\vec{a}$,$\overline{u}$) **:**
24 **for** *each agent* $i \in \mathcal{N}$ **do**
25 $N_i(s, a_i) = N_i(s, a_i) + 1$;
26 **for** *each utility function* $u_j \in \mathcal{U}$ **do**
27 $N_{u_j}(s, \vec{a}_{u_j}) = N_{u_j}(s, \vec{a}_{u_j}) + 1$;
28 $q_{u_j}(s, \vec{a}_{u_j}) = \sum_{i \in Neg_{u_j}(s)} \frac{q_i(s, a_i)}{|Neg_i(s)|}$;
29 $\overline{u}_j(s, \vec{a}_{u_j}) = \overline{u}_j(s, \vec{a}_{u_j}) + \frac{q_{u_j}(s, \vec{a}_{u_j}) - \overline{u}_j(s, \vec{a}_{u_j})}{N_{u_j}(s, \vec{a}_{u_j})}$.

UPDATESTATE (Steps 23-29): Before updating each agent and each utility function, in Step 10, we need to take $O(n2^k)$ to compute the Shapley values of all agents (i.e., Eq. (7)). Then, we need to take $O(n)$ and $O(kn)$ computations to update visit frequency $N(s, a_i)$ and $N_{u_j}(s, \vec{a}_{u_j})$, respectively. In total, the function UPDATESTATE will take $O(n+kn+n2^k) = O(n2^k)$ computations.

In summary, each simulation of Algorithm 2, the total computations of the three functions INITIALIZESTATE, MAX-SUM, and UPDATESTATE is $O(d^2n^2 + Tk^2d^k + n2^k)$. □

Next, we prove that we can preserve and backpropagate the global payoffs using our coordinated MCTS.

Lemma 4: Let $\rho = \langle s_0, s_1, \cdots, s_T \rangle$ denote the sequence of states visited by the simulated global joint-action $\langle \vec{a}_0, \vec{a}_1, \cdots, \vec{a}_T \rangle$. The cumulative global reward of ρ is $V^\rho(s_0, \vec{a}_0) = \sum_{k=0}^{T-1} \gamma^k R(s_k, \vec{a}_k, s_{k+1})$, where $R(s_k, \vec{a}_k, s_{k+1})$ is the global payoff of the global joint-action $\vec{a}_k$ at state s_k. At the root state s_0, let $\mathcal{G}(s_0) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ denote the DCOP model and $q_{u_j}(s_0, \vec{a}_{0,u_j})$ denote the cumulative value of the local utility function u_j obtained from ρ using our coordinated MCTS. Then, we have

$$V^\rho(s_0, \vec{a}_0) = \sum_{u_j \in \mathcal{G}(s_0)} q_{u_j}(s_0, \vec{a}_{0,u_j}). \tag{13}$$

Proof: Based on the efficiency of the SV-SCA mechanism (that is, Lemma 1), we have

$$\begin{aligned} V^\rho(s_0, \vec{a}_0) &= \sum_{k=0}^{T-1} \gamma^k R(s_k, \vec{a}_k, s_{k+1}) \\ &= \sum_{k=0}^{T-1} \gamma^k \sum_{i \in \mathcal{N}} R_i(s_k, \vec{a}_k, s_{k+1}) \qquad (14) \\ &= \sum_{k=0}^{T-1} \sum_{i \in \mathcal{N}} \gamma^k R_i(s_k, \vec{a}_k, s_{k+1}) = \sum_{i \in \mathcal{N}} q_i(s_0, a_{0,i}). \qquad (15) \end{aligned}$$

Eq. (14) holds since $R(s_k, \vec{a}_k, s_{k+1}) = \sum_{i \in \mathcal{N}} \phi_i^{\vec{a}_k} = \sum_{i \in \mathcal{N}} R_i(s_k, \vec{a}_k, s_{k+1})$. According to Step 27 in Algorithm 2, the cumulative value of local utility u_j is the sum of the individual cumulative rewards $q_i(s_0, a_{0,i})$ of the neighboring agent $i \in Neg_{u_j}$.

$$\begin{aligned} \sum_{u_j \in \mathcal{G}(s_0)} q_{u_j}(s_0, \vec{a}_{0,u_j}) &= \sum_{u_j \in \mathcal{G}(s_0)} \sum_{i \in Neg_{u_j}(s_0)} \frac{q_i(s_0, a_{0,i})}{|Neg_i(s)|} \\ &= \sum_i q_i(s_0, a_{0,i}) = V^\rho(s_0, \vec{a}_0). \end{aligned} \tag{16}$$

Therefore, we can conclude that our coordinated MCTS can preserve and backpropagate the global payoff through an individual agent's reward. □

VI. SV-SCA Guided Coordinated MARL

In this section, under the guidance of SV-SCA, we propose a fully-decentralized and coordinated MARL for MMDPs without access to environmental dynamics.

A. Basic MARL Approaches

Deep Q-learning, which uses neural networks as approximations, is a popular learning approach for solving single-agent decision-making problems. Let $Q(s, a)$ represent the expected cumulative reward for action a in the state s. The policy π can be derived from $Q(s, a)$ by setting $\pi(s) = \arg\max_{a \in A} Q(s, a)$.

1) Centralized MARL: Centralized MARL assumes that a central controller can gather all global information including global joint-actions and global rewards. During training, the centralized MARL can directly estimate the global state-action value $Q_{tot}(s, \vec{a})$ using the standard single-agent Q-learning algorithm:

$$\begin{aligned} Q_{tot}^{t+1}(s, \vec{a}) = (1-\alpha) Q_{tot}^t(s, \vec{a}) + \alpha [R(s, \vec{a}, s') \\ + \gamma \max_{\vec{a}'} Q_{tot}^t(s', \vec{a}')]. \end{aligned} \tag{17}$$

where $\alpha \in (0, 1)$ is the learning rate; $R(s, \vec{a}, s')$ is the immediate global reward for taking $\vec{a}$ in state s; $\gamma \in [0, 1]$ is a discount factor. Although this centralized approach can produce an optimal policy, it is practically intractable because the global joint-action space is exponential in terms of the number of agents.

2) Independent MARL: At the other extreme, we can use an independent learning approach [62] in which the agents ignore the exploration behavior of other agents and learn the state-action value function $Q_i(s, a_i)$ only from their local reward $R_i(s, a_i, s')$. The local reward of each agent i can be provided by the global reward $R(s, \vec{a}, s')$ or difference reward $R(s, \vec{a}, s') - R(s, \vec{a}_{-i}, s')$ [17], [18]. An independent MARL is distributed, resulting in computational savings in the policy space. However, this approach is not coordinated and may convergence to policies that are locally but not globally optimal.

B. Coordinated MARL

Coordinated MARL attempts to achieve both scalability and optimality (or near optimality) using structural interactions and coordinated distributed learning. Existing explicit coordinated MARL approaches [63], [64], [65], [66] use individual rewards directly to update local joint-action values. Existing implicit coordinated MARL approaches [16], [54] learn to decompose the global state action value Q_{tot} into individual state action values Q_i using neural networks. In contrast to the implicitly coordinated MARL, our coordinated MARL based on SV-SCA does not require learning the global state action Q_{tot}, which is a more flexible fully-decentralized learning paradigm [59]. On the other hand, in comparison with the explicitly coordinated MARL based on difference reward, we leverage the SV-SCA to redistribute the global reward fairly to individual rewards.

1) Global Q-Value Decomposition in DCOP: Given the interaction structure, DCOP $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ at state s, the coordinated MARL optimizes the decomposable Q-value $\hat{Q}_{tot}$, which is used to approximate the real global Q-value Q_{tot}. The approximator $\hat{Q}_{tot}$ can be defined as the sum of the local values Q_{u_j}:

$$\hat{Q}_{tot}(s, \vec{a}) = \sum_{u_j \in \mathcal{U}} Q_{u_j}(s, \vec{a}_{u_j}), \tag{18}$$

where the value $Q_{u_j}(s, \vec{a}_{u_j})$ is the expected cumulative reward of the agents in the utility function u_j by performing a joint-action $\vec{a}_{u_j}$ at state s, and then behaving globally optimally with respect to maximizing $\hat{Q}_{tot}(s, \vec{a})$. By decomposing Eq. (18), the global Q-learning update rule for $\hat{Q}_{tot}(s, \vec{a})$ in Eq. (17) can be rewritten as

$$\hat{Q}_{tot}^{t+1}(s, \vec{a}) = \sum_{u_j \in \mathcal{U}} Q_{u_j}^{t+1}(s, \vec{a}_{u_j}) = (1 - \alpha) \sum_{u_j \in \mathcal{U}} Q_{u_j}^{t}(s, \vec{a}_{u_j}) + \alpha \left[R(s, a, s') + \gamma \max_{\vec{a}'} \sum_{u_j \in \mathcal{U}} Q_{u_j}^{t}(s', \vec{a}'_{u_j}) \right]. \tag{19}$$

To provide distributed and coordinated learning for MARL, two issues must be addressed to update the state action value Q_{u_j} of the local utility functions u_j. The first is to redistribute the global reward $R(s, \vec{a}, s')$ to the local utility function and the second is to determine the coordinated global joint-action $\vec{a}'$ that maximizes the expected value $\sum_{u_j \in \mathcal{U}} Q_{u_j}^{t}(s', \vec{a}'_{u_j})$. We address these two issues using the following techniques:

- **SV-SCA for Decomposing the Global Reward $R(s, \vec{a}, s')$.** At state s, let $\vec{a} = (a_1, \cdots, a_n)$ denote the global joint-action executed by agents. We first use the SV-SCA (that is, Eq. (1)) to determine the immediate individual reward for each agent i, that is, $\forall i, R_i(s, a_i, s') = \phi_i^{\vec{a}}(s, \vec{a}, s')$. Then, the immediate reward of the local utility function u_j is defined as the sum of the weighted individual rewards of neighbor agents $i \in Neg_{u_j}(s)$, that is,
$$R_{u_j}(s, \vec{a}_{u_j}, s') = \sum_{i \in Neg_{u_j}(s)} \frac{R_i(s, a_i, s')}{|Neg_i(s)|}, \tag{20}$$
where $Neg_{u_j}(s)$ denotes the set of neighbor agents of u_j and $Neg_i(s)$ denotes the set of neighbor utility functions of i in the current DCOP $\mathcal{G}(s)$.
- **Max-sum(0) for Maximizing $\sum_{u_j \in \mathcal{U}} Q_{u_j}^{t}(s', \vec{a}'_{u_j})$.** We can directly use Max-sum(0) (that is, Algorithm 1) to compute the optimal global joint-action $\vec{a}'$ at state s'.

Both SV-SCA for decomposing $R_{u_j}(s, \vec{a}, s')$ and maximizing $\sum_{u_j \in \mathcal{U}} Q_{u_j}^{t}(s', \vec{a}'_{u_j})$ using Max-sum(0) can be executed in a fully distributed manner. Now, Eq. (19) can be decomposed into local Q_{u_j} values for each utility function u_j:

$$Q_{u_j}^{t+1}(s, \vec{a}_{u_j}) = (1 - \alpha) Q_{u_j}^{t}(s, \vec{a}_{u_j}) + \alpha \Big[R_{u_j}(s, \vec{a}_{u_j}, s') + \gamma Q_{u_j}^{t} \left(s', \vec{a}'_{u_j} | \vec{a}'_{u_j} \in \arg\max_{\vec{a}'} \sum_{u_j \in \mathcal{U}} Q_{u_j}^{t}(s', \vec{a}') \right) \Big]. \tag{21}$$

Finally, the coordinated MARL based on SV-SCA is formally presented in Algorithm 3, and Fig. 3 shows the coordinated MARL framework.

C. Theoretical Property

The monotonicity property has proven to be critical for promoting coordination in MARL, in which each agent optimizes its own rewards also optimize the system's global reward [16], [18]. In this section, we demonstrate the monotonic property of the coordinated MARL based on SV-SCA.

Theorem 2: Let $Q_{tot}(s, \vec{a}) = \mathbb{E}_\pi \left[\sum_{k=0}^{T-1} \gamma^k R(s_k, \vec{a}_k, s_{k+1}) | s_0 = s, \vec{a}_0 = \vec{a} \right]$ denote the expected cumulative global reward under the global joint policy $\pi = \times_{1 \le i \le n} \pi_i$, and $Q_{u_j}(s, \vec{a}_{u_j}) = \mathbb{E}_\pi \left[\sum_{k=0}^{T-1} \gamma^k R_{u_j}(s_k, \vec{a}_{u_j,k}, s_{k+1}) | s_0 = s, \vec{a}_{u_j,0} = \vec{a}_{u_j} \right]$ denote the cumulative reward of local utility function $u_j \in \mathcal{U}$. Assume that the utility function $u(\cdot)$ is differentiable, the SV-SCA based coordinated MARL satisfies the monotonic property:

$$\frac{\partial Q_{tot}(s, \vec{a})}{\partial Q_{u_j}(s, \vec{a}_{u_j})} \ge 0, \forall u_j \in \mathcal{U}. \tag{22}$$

Proof: Since $Q_{tot} = \sum_{k=0}^{T-1} \gamma^k R(s_k, \vec{a}_k, s_{k+1})$ and $Q_{u_j} = \sum_{k=0}^{T-1} \gamma^k R_{u_j}(s_k, \vec{a}_{u_j,k}, s_{k+1})$, we only need to prove $\frac{\partial R(s, \vec{a}, s')}{\partial R_{u_j}(s, \vec{a}_{u_j}, s')} \ge$

Algorithm 3 SV-SCA Based Coordinated MARL

Input : The DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$.

Output: The estimated value $Q_{u_j}(s, \vec{a}_{u_j}, \theta_j)$ for each utility function u_j.

for *episode=1,2,* $\cdots$ **do**
  Initialize the state s
  for t*=1,2,* $\cdots$ **do**
    Each agent i computes an individual action a_i^* by Max-sum (0) and executes the individual action a_i^* in an ϵ-greedy manner. Each agent i computes the individual reward $R_i(s, \vec{a}) = \phi_i^{\vec{a}}(s)$ using Eq. (1);
    The system transitions to the next state, s'. Store the tuple $\langle s, \vec{a}^*, \phi^{\vec{a}^*}, s' \rangle$ in the replay buffer $\mathcal{P}$;
    for $u_j \in U$ **do**
      $y_{u_j} = \sum_{i \in Neg_{u_j}} \frac{R_i(s,\vec{a},s')}{|Neg_i(s)|}$
        $+\gamma\, Q^t_{u_j}(s', \vec{a}'_{u_j} | \vec{a}'_{u_j} \in$
          $\arg\max_{\vec{a}'} \sum_{u_j \in U} Q_{u_j}(s', \vec{a}', \theta_{u_j}))$;
      Minimize the loss function $L(\theta_{u_j}) = \left(y_{u_j} - Q_{u_j}(s, \vec{a}_{u_j}, \theta_{u_j})\right)^2$ to update the network parameter θ_{u_j};
      Update the target network $\theta'_{u_j} = \tau\theta_{u_j} + (1-\tau)\theta'_{u_j}$.

0 at each state s. Particularly, under the SV-SCA, the reward $R_i(s, a_i, s')$ of each agent i satisfies Eq. (23), as shown at the bottom of the next page. In Eq. (23), since the coalition $\mathcal{N}\backslash\{\mathcal{C} \cup i\})$ does not include the agent i, the utility function $u(\mathcal{N}\backslash\{\mathcal{C} \cup i\})$ of this coalition is independent of the action a_i of agent i. For each utility function $u_j \in \mathcal{U}$, the reward $R_{u_j}(s, \vec{a}_{u_j}, s') = \sum_{i \in Neg_{u_j}} \frac{R_i(s,a_i,s')}{|Neg_i(s)|}$, we further have that

$$\begin{aligned}
&\frac{\partial R_{u_j}(s, \vec{a}_{u_j}, s')}{\partial \vec{a}_{u_j}} \\
&= \frac{\partial \sum_{i \in Neg_{u_j}} \frac{R_i(s,a_i,s')}{|Neg_i(s)|}}{\partial \vec{a}_{u_j}} \\
&= \frac{\sum_{i \in Neg_{u_j}} \frac{1}{|Neg_i(s)|} \partial R_i(s, a_i, s')}{\partial a_i \partial \vec{a}_{u_j - i}} \\
&= \frac{\sum_{i \in Neg_{u_j}} \frac{1}{|Neg_i(s)|} \sum_{\mathcal{C} \subseteq \mathcal{N}\backslash\{i\}} \frac{|\mathcal{C}|!(n-|\mathcal{C}|-1)!}{n!} \partial u(\mathcal{N})}{\partial \vec{a}_{u_j}}
\end{aligned} \tag{24}$$

At the state s, since $u(\mathcal{N})$ and $R(s, \vec{a}, s')$ both denote the immediate global reward (that is, $u(\mathcal{N}) = R(s, \vec{a}, s')$), we can conclude that $\frac{\partial R(s,\vec{a},s')}{\partial R_{u_j}(s,\vec{a}_{u_j},s')} = \frac{\frac{\partial R(s,\vec{a},s')}{\partial a_{u_j}}}{\frac{\partial R_{u_j}(s,\vec{a}_{u_j},s')}{\partial a_{u_j}}} \geq 0$, which will satisfy $\frac{\partial Q_{tot}(s,\vec{a})}{\partial Q_{u_j}(s,\vec{a}_{u_j})} \geq 0$. □

VII. Experiments

In this section, we validate the proposed SV-SCA-based coordinated MCTS for online MAP and SV-SCA-based coordinated MARL. All computations are performed on a 64-bit workstation with 64 GB RAM and a 16-core 3.5 GHz processor. All records are averaged over 40 instances and standard errors are used as confidence intervals.

A. Experiments for Online MAP

1) Experiment Domains: We introduce three typical domains that can represent a range of sequential multiagent coordination problems with static and dynamic DCOPs.

a) Traffic Signal Control (TSC) [67]: We evaluate the proposed methods for the TSC problem on the Cityflow simulation platform [67] within real-world and synthetic traffic networks (as shown in Fig. 4(a)). Two synthetic grid-like traffic networks, syn_3 × 3 and syn_4 × 4, are generated by the Cityflow. Two representative real-world traffic datasets, jinan _3 × 4 and hangzhou _4 × 4, are collected from the two cities [68]. In each traffic network, each vehicle is described by (o, t, d), where o is the origin location (that is, link), t is time, and d is the destination location. Locations o and d are links in the road network. Each intersection is modeled as an agent, the links between agents are modeled as local interactions in DCOP, and the interaction structure is static. The ultimate objective of TSC is to minimize the average travel time of vehicles. However, the travel time of a vehicle is a long-term object that depends on the sequence of signal actions that cannot be computed until the vehicle has completed its route. In our coordinated MCTS, we use the number of vehicles that exit the traffic network in the planning step as the immediate global payoff.

b) Compared Methods: We compare our coordinated MCTS method to the following four baselines:

- **Maxpressure** [69], a traditional transportation method, which greedily actives the phase with the maximum pressure.
- **PressLight** [28], an independent MARL method where each agent learns the policy of selecting the next phase by vanilla deep Q-network (DQN).
- **FV-MCTS** [46], where the individual action is evaluated according to the global payoff.
- A variant of our coordinated MCTS with lookahead depth=1.

c) Multi-Robot Patrolling (MRP) [70]: In MRP, multiple mobile robots are deployed to patrol sensitive regions, where persistent surveillance, inspection, and control are required (as shown in Fig. 4(b)). The MRP can be formulated as a graph $G = \langle V, E \rangle$, where V is the set of regions to be patrolled and $e_{ij} \in E$ denotes the regions v_i and v_j are adjacent. The robots can navigate between adjacent regions. The instantaneous idleness of a region in the current period is the number of periods that have elapsed since its last visit. In each period, the two robots interact locally if they can move to the same region in the next period, that is they have a joint effect on the idleness of a region. In MRP, the interaction structure is dynamic. The objective of MRP task is to coordinate the patrolling strategies of the robots to minimize the average idleness of all regions over the entire horizon.

d) Compared Methods: In addition to the **FV-MCTS**, we compare our coordinated MCTS with 1) **Naive**

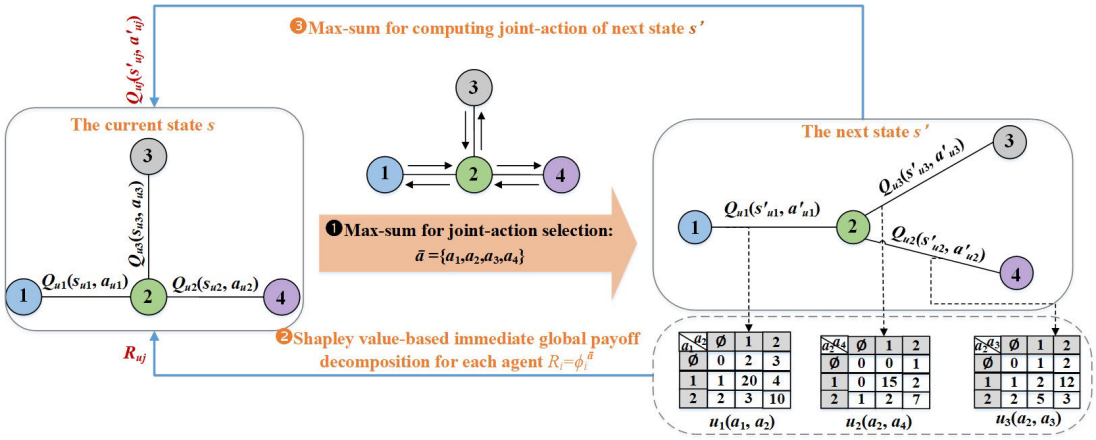
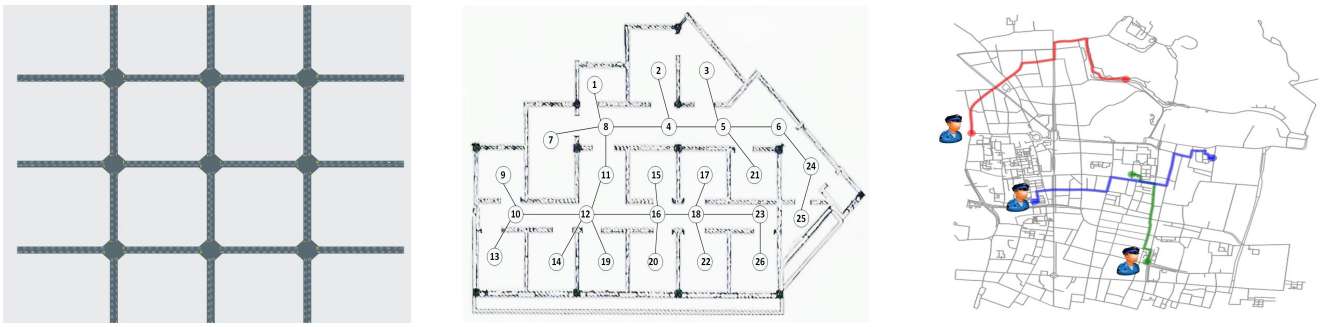


Fig. 3. Overview of coordinated MARL. All of the modules ━, ➗ and ↘ can be computed in a fully-decentralized manner.



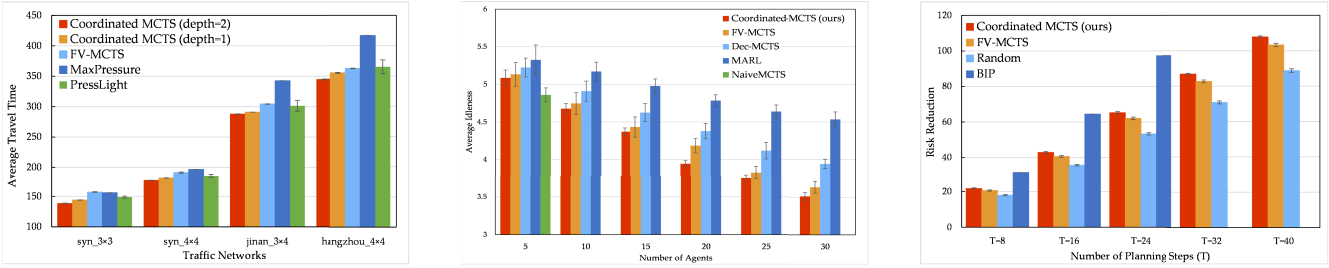
(a) Traffic Signal Control (TSC) (b) Multi-Robot Patrolling (MRP) (c) Security Traffic Patrolling (STP)

Fig. 4. Multiagent planning domains. **LEFT:** the TSC domain where each point indicates an intersection and each intersection is modeled as an agent, and the links between agents are modeled as the local interaction in DCOP, **MIDDLE:** the MRP domain where two robots can interact with each other locally if they have an effect on the same neighbor regions with respect to idleness, which can be modeled the interaction structure in DCOP, and **RIGHT:** the STP domain where two police officers can interact with each other locally if they have an effect on the same neighbor region with respect to the risk of accidents, which can be modeled the interaction structure in DCOP.

MCTS, where the global joint-action is explored and evaluated, and 2) independent **MARL** [70], where each agent adopts model-free Q-learning to learn the patrolling policy.

e) Security Traffic Patrolling (STP) [71]: In STP, a set of police officers (that is, agents) are deployed for traffic enforcement with the aim of preventing road accidents or illegal driver behaviors (as shown in Fig. 4(c)). The interaction between police and drivers can be cast as a defender-attacker Stackelberg game. The defender (police) commits to a pure patrol strategy used to generate daily patrol schedules for each police officer. A daily patrol schedule comprises a trajectory through the road network, that is, a sequence of regions to patrol. The attacker (that is, driver) follows the opportunistic behavior model and reacts to police enforcement in the current and past periods. The risk of accidents measures the likelihood of serious traffic accidents occurring in each region during each period. The STP problem focuses on coordinating police officers' patrolling strategies with the aim of minimizing the risks of accidents occurring throughout the game. In STP, the interaction structure is similar to that in MRP and is dynamic and state-dependent.

$$
\begin{aligned}
&\frac{\partial R_i(s,a_i,s')}{\partial a_i}\\
&=\frac{\partial\sum_{\mathcal{C}\subseteq\mathcal{N}\setminus\{i\}}\frac{|\mathcal{C}|!(n-|\mathcal{C}|-1)!}{n!}\triangle_i^{\vec{a}}(\mathcal{C})}{\partial a_i} && \triangleright\text{The Shapley value}\\
&=\frac{\partial\sum_{\mathcal{C}\subseteq\mathcal{N}\setminus\{i\}}\frac{|\mathcal{C}|!(n-|\mathcal{C}|-1)!}{n!}u(\mathcal{C}\cup i)}{\partial a_i} && \triangleright\triangle_i^{\vec{a}}(\mathcal{C})=u(\mathcal{C}\cup\{i\})-u(\mathcal{C})\\
&=\frac{\partial\sum_{\mathcal{C}\subseteq\mathcal{N}\setminus\{i\}}\frac{|\mathcal{C}|!(n-|\mathcal{C}|-1)!}{n!}[u(\mathcal{N})-u(\mathcal{N}\setminus\{\mathcal{C}\cup i\})]}{\partial a_i} && \triangleright\text{The Efficiency Property}\\
&=\sum_{\mathcal{C}\subseteq\mathcal{N}\setminus\{i\}}\frac{|\mathcal{C}|!(n-|\mathcal{C}|-1)!}{n!}\frac{\partial u(\mathcal{N})}{\partial a_i}
\end{aligned}
\tag{23}
$$



(a) Results on TSC (b) Results on MRP (c) Results on STP

Fig. 5. Experimental results. **LEFT:** in TSC, methods are evaluated on average travel time of vehicles (the lower the better), **MIDDLE:** in MRP, methods are evaluated on average idleness of nodes (the lower the better), and **RIGHT:** in STP, methods are evaluated on reducing risks of accidents (the higher the better). Methods that exceeded the timeout (that is, 10 minutes) do not appear.

f) Compared Methods: In addition to online baselines **Naive MCTS** and **FV-MCTS**, we also compare our coordinated MCTS against a randomized patrolling policy (**Random**) and **binary integer programming (BIP)** [71]. BIP is used to formulate the pure strategy for traffic enforcement and a master/slave-based optimization is used for scale up.

2) Experiment Results:

a) Test the Efficiency: Fig. 5 shows the efficiency of our coordinated MCTS in improving the system performance in different domains.

In the TSC domain, we evaluate the methods on two synthetic networks, syn_3×3 and syn_4×4, and two real-world road networks, jinan_3×4 and hangzhou_4×4. The performance is evaluated based on the average travel time of the vehicles. From Fig. 5(a), we find that in all networks, compared with online MAP and offline RL baselines, our coordinated MCTS method generates the shortest average travel time. The potential reason for this is that for complex road networks with a number of intersections, it is difficult for RL methods to train desirable coordination solutions. By contrast, our method enables coordination between neighboring intersections and aims to optimize global traffic. However, MaxPressure only optimizes the local pressure at individual intersections; thus, its performance decreases in complex traffic situations. Moreover, within the time limits, it is better to use a longer planning horizon (that is, coordinated MCTS depth= 2) than a shorter planning horizon (that is, coordinated MCTS depth= 1).

In the MRP domain, we evaluate the methods on a synthetic 16×16 grid-like graph and varied the number of agents (that is, robots) between 5 and 30. The performance is evaluated based on the average idleness of the nodes. Fig. 5(b) shows that our coordinated MCTS can achieve the minimum average idleness, which guarantees that sensitive nodes can be patrolled frequently. When there are a smaller number of agents (that is, five), Naive MCTS performs better than FV-MCTS. This can be explained by the fact that a Naive MCTS can explore promising global joint actions when the number of agents is small. However, the naïve MCTS reaches a timeout even in problems with ten agents.

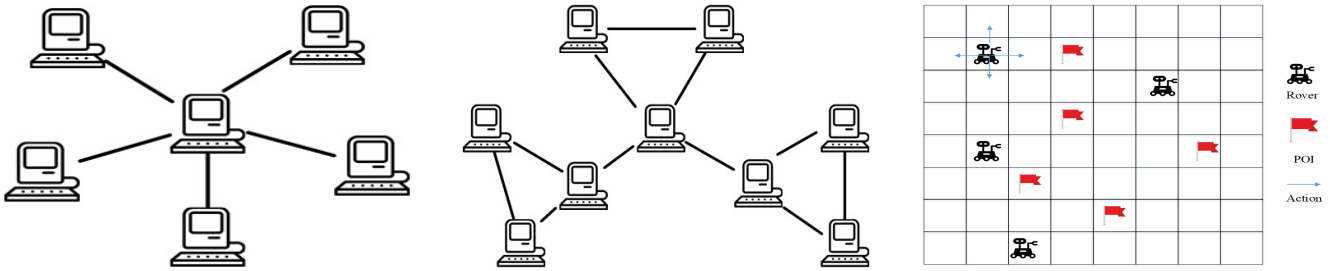
In the STP domain, we evaluate the methods using a real-world road network comprising 284 intersections and 355 roads. The number of agents (that is, police officers) is set to 30 and the number of planning steps (that is, T) varied between 8 and 40. The performance is evaluated based on the risk (of accidents) reduction between the no-enforcement condition and the proposed methods. From Fig. 5(c), we find that, compared with FV-MCTS, our coordinated MCTS can reduce the no-enforcement risk value by more than 5%. Because BIP always returns optimal solutions, BIP can reduce the largest risks. However, the complexity of BIP increases exponentially with the planning steps (T), preventing its application to larger problems (for example, $T \geq 32$).

b) Test the Scalability: In the TSC domain (that is, Fig. 5(a)), because the traffic changes in seconds, we set the time limit for online computations to 3s. In all traffic networks, the proposed coordinated MCTS returns the best solution when executed online. In the MRP and STP domains, because agents must patrol a region for some time, we can compute online plans for the next step during patrolling. Thus, in the MRP and STP domains, we set the time limit for online computations to 60s. Fig. 5(b) and 5(c) show the scalability of our coordinated MCTS with respect to the number of agents n and planning steps (T), and we find that our coordinated MCTS and FV-MCTS can be applied to large-scale problems with tens of agents (that is, $n = 30$) and long planning steps (that is, $T = 40$). In comparison, Fig. 5(b) shows that the naïve MCTS cannot be applied to problems with more than ten agents, and Fig. 5(c) shows that BIP cannot be applied to problems with more than 32 planning steps.

c) Test the Computation Time: Table II shows the computation time of our coordinated MCTS in the TSC domain. From Table II, it can be found that 1) the computation time for the simulation step increases linearly with the number of agents, and 2) given the limited time budget (that is, #simulation= 500), our coordinated MCTS with depth= 2 performs better (with respect to ATT) than the variant with depth= 1, but nearly the same with that of depth= 3. In conclusion, the depth setting depends on both the time budget and number of agents. For a small budget and large number of agents, a short depth is preferred, whereas for a large budget and a small number of agents, a long depth is better.

B. Experiments for MARL

1) Experiment Domains: In the MARL tasks, we introduce two additional domains with static and dynamic coordination graphs.



(a) Star (b) Ring-of-Rings (c) Multi-Rover Domain

Fig. 6. Multiagent reinforcement learning domains. **LEFT** and **MIDDLE**: Two typical network topology of SysAdmin problems. In each network, the action of each machine has an effect on neighbor's status. **RIGHT:** the Muti-Rover problem where multiple rovers attempt to observe points of interest (POIs) on a grid plane.

TABLE II

TEST RESULTS OF THE COMPUTATION TIME OF OUR COORDINATED MCTS IN THE TSC DOMAIN; THE TOTAL NUMBER OF ITERATIONS IS 500. **ST:** THE COMPUTATION TIME (MS) FOR EACH SIMULATION STEP IN ALGORITHM 2, AND **ATT:** THE AVERAGE TRAVEL TIME OF VEHICLES

Traffic Network		Coordinated MCTS		
		depth=1	depth=2	depth=3
syn_3 × 3	ST (ms)	10.3	10.9	11.1
	ATT	144.1	138.4	138.4
syn_4 × 4	ST (ms)	19.8	20.5	20.4
	ATT	180.3	176.4	176.1
jinan_3 × 4	ST (ms)	13.4	14.4	16.3
	ATT	290.9	287.6	287.2
hangzhou_4 × 4	ST (ms)	19.3	20.5	21.0
	ATT	355.0	344.7	344.4

a) The SysAdmin with Static DCOP [46], [72]: The SysAdmin domain is used as the standard MMDP benchmark to maintain a network of n machines. In SysAdmin, each agent represents a machine and is connected to a set of other machines. Thus, the interaction structure in SysAdmin is as static as the network topology. Fig. 6(a) and Fig. 6(b) illustrate two typical network topologies for SysAdmin: star and ring-of-rings for SysAdmin. In SysAdmin, each machine is associated with two state variables: Status $S_i \in \{\text{GOOD}, \text{FAULTY}, \text{DEAD}\}$ and Load $L_i \in \{\text{IDLE}, \text{LOADED}, \text{SUCCESS}\}$. A GOOD machine can become FAULTY with a certain probability $p_{\text{GOOD}\to\text{FAULTY}}$, and a FAULTY machine becomes DEAD with the probability $p_{\text{FAULTY}\to\text{DEAD}}$. A DEAD machine increases the probability of its neighbor dying. These processes arrive at machines using a static but unknown distribution. A machine is IDLE when there is no process, and LOADED otherwise. The system receives a reward of one if a process terminates successfully; processes take longer when the status is FAULTY, and a DEAD machine loses the process. Each agent must decide whether to reboot its machine, in which case the status becomes GOOD; however, any running process will be lost.

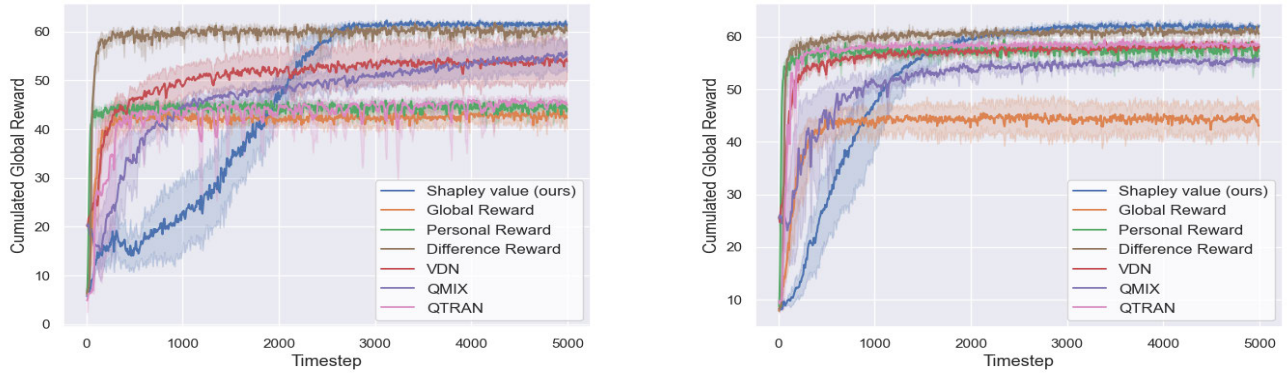
b) The Multi-Rover with Dynamic DCOP [73]: In the multirover problem (Fig. 6(c)), multiple rovers attempt to observe points of interest (POIs) on a grid plane. At each step, each rover can perform one of the following actions {UP, DOWN, RIGHT, LEFT, or STAY}. A POI j has a fixed position on the plane, and a value V_j is associated with it. The value of the information obtained by observing a POI is inversely related to the straight-line distance of the rover from the POI. Although any rover can observe any POI in terms of the global reward, only the closest observation counts. The global reward G_t is given by the aggregate value of all the POIs observed at each step t; that is,

$$G_t = \sum_j \frac{V_j}{\min_{i\in\mathcal{N}} \delta(L_j, L_i)}, \tag{25}$$

where L_j is the location of the POI j, L_i is the location of rover i, and $\delta(L_j, L_i)$ denotes the straight-line distance between rover i and POI j. Thus, two rovers i and j are connected in DCOP if there exists on POI k, such that $|\delta(L_i, L_k) - \delta(L_j, L_k)| \le 1$. In this study, we set the number of rovers to eight, the number of POIs to 14, and the plane comprised 50×50 grids. In this multirover domain with a dynamic DCOP, learning the local Q-value of each utility function is inefficient. Instead, we directly exploit the SV-SCA to determine the immediate reward of each agent i, which is used to learn its individual value Q_i.

c) Compared Methods: We compare our SV-SCA based coordinated MARL (that is, **Shapley value (ours)**) method with six baselines:

- Independent MARL with **Personal Reward** [46], which is an explicit SCA-based MARL and each agent uses its individual reward to update its individual Q-value.
- Independent MARL with **Global Reward** [74], which is an explicit SCA-based MARL and each agent uses the global reward to update its individual Q-value.
- Independent MARL with **Difference Reward** [18], [31], which is also an explicit SCA-based MARL using the marginal difference reward to update its individual Q-value.
- **COMA** [19], which proposes the difference critic for agents, i.e., the difference between the global Q-value and counterfactual global without Q-value this agent. **SHAQ** [33], which proposed the Shapley value as an alternative critic Q_i^ϕ for each agent, where Q_i^ϕ is computed as the average contribution of the marginal state-action value $Q_\mathcal{C}$ to any coalition $\mathcal{C}$.
- **VDN** [15], which is an implicit SCA and the neural network constrains the sum of the individual Q_i value within the global Q_{tot} value.



(a) Results on Star

(b) Results on Ring-of-Rings

Fig. 7. The efficiency of our coordinated MARL based on SV-SCA on improving system performance in SysAdmin with static DCOP.

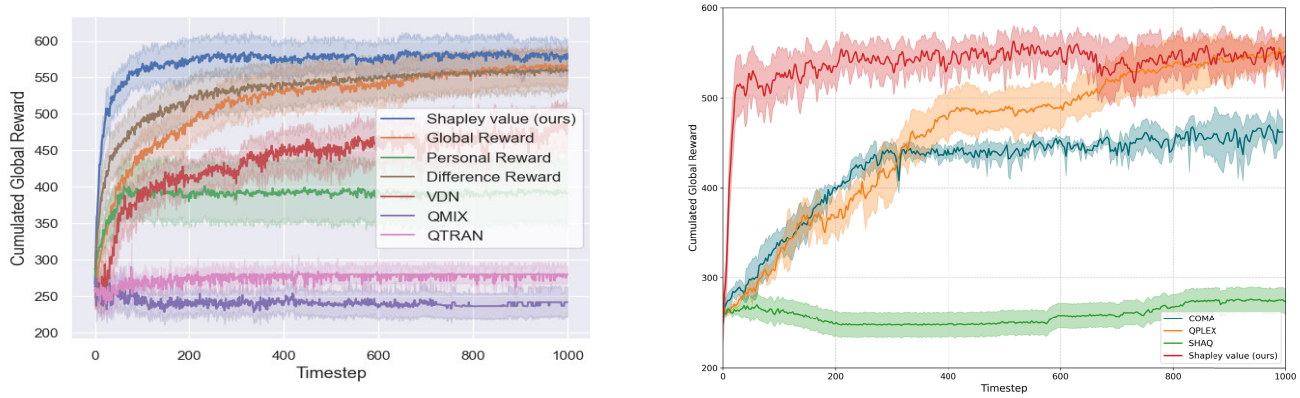


Fig. 8. The efficiency of our coordinated MARL based on SV-SCA on improving system performance in multi-rover with dynamic DCOP.

- **QMIX** [16], which is an implicit SCA and the neural network constrains the monotonicity relationship between each individual Q_i and the global Q_{tot} value.
- **QTRAN** [29], which is also an implicit SCA and learns the individual-global-max (IGM) relationship between each individual Q_i and the global Q_{tot} value by neural networks.
- **QPLEX** [30], which introduces the dueling structure $Q_{tot} = V_{tot} + A_{tot}$ for representing both joint and individual action-value functions and then reformalizes the IGM principle as an advantage-based IGM.

d) Experiment Results: Fig. 7 and Fig. 8 show the efficiency of our coordinated MARL based on SV-SCA for improving system performance. From Figs. 7(a) and (b) with a static DCOP, we can observe that our coordinated MARL performs the best, followed by the difference reward. The potential reason for this is that when the DCOP is static, the explicit SCA (that is, difference reward) can determine the contribution of individual agents efficiently. By contrast, MARL methods with implicit SCA struggle to learn the decomposition relationships even with a large number of experiences. From Fig. 8 with a dynamic DCOP, we can observe that our coordinated MARL achieves a significantly larger reward than those of these benchmarks. This result can be explained by the fact that 1) compared to the existing explicit SCA (that is, difference reward, global reward, personal reward and COMA), SV-SCA not only has a monotonic property, but also satisfies the additivity property. 2) compared to SHAQ that requires a large number of experience to learn the global Q-value, our coordinated MARL requires far fewer experience to learn its individual policy based on its own critic. 3) Compared to these implicit SCA methods that satisfy monotonicity (that is, VDN), additivity property (that is, QMIX), or IGM property (that is, QTRAN and QPLEX), our SV-SCA is explicit and can be computed exactly through the combination of DCOP.

In summary, our coordinated MARL based on SV-SCA outperforms the existing MARL benchmarks for both static and dynamic DCOPs.

VIII. Discussion and Limitations

A. On Model-Free Environments Without Explicit Coordination Structure

In this paper, the proposed MAP and MARL are model-based methods where the underlying interaction structure DCOP is known. Moreover, this paper mainly focuses on the multiagent Markov decision process (MMDP) problem, where each agent can observe the global state, but only depends on actions of neighbor agents.In the model-free environments where the interaction structure is unknown, MMDP becomes the decentralized partial observable Markov decision

process (Dec-POMDP). For Dec-POMDP problems, we can adopt the centralized-training-decentralized execution (CTDE) framework to train our proposed SV-SCA algorithm [75]. In centralized training phase, we can integrate relevant information across all agents, and learn a self-attention network to determine an implicit coordination structure between agents [55]. The coordination structure can be used for computing Shapley value as well as the critic for each agent. Based on the individual critic, the policy of each agent can be trained using the standard actor-critic method depends on the local observability. During the execution phase, the centralized self-attention network is no longer required and the agents can act independently based on the local observability.

B. Scalability of Shapley Value in Large-Scale Multiagent Scenarios

In the case of large-agent scenarios (e.g., $n = 1000$ and $n_i^{loc} > 10$), computing $2^{n_i^{loc}}$ is still challenging. There are two approximation methods for mitigating the computational burden. 1) The approximated Shapley value can be obtained by Monte Carlo sampling [24]. For example, given the total number of coalitions $2^{n_i^{loc}}$ to be considered for computing the Shapley value, we can only use the M Monte Carlo samplings from $2^{n_i^{loc}}$. Based on this approximation, the computational complexity of the Shapley value can be reduced from $O(2^{n_i^{loc}})$ to $O(M)$, where M is a hyperparameter that represents the number of Monte Carlo sampling and can be a tuned to tradeoff Shapley-value efficiency and effectiveness. 2) We can also reduce the number of neighbor agents n_i^{loc} (e.g., $n_i^{loc} \leq 10$). A self-attention network can be trained to learn the implicit coordination structure between agents [55]. We can tune the hyperparameter of soft edge weights to regulate the number of neighbor agents n_i^{loc}. For example, a large threshold of edge weights might induce sparse coordination structure with small n_i^{loc}, vice versa. The Shapley value then can be computed based on the implicit interaction structure learned by the self-attention network.

IX. Conclusion

By combining SV with DCOP, a novel SCA mechanism that can satisfy both the efficiency and monotonicity properties is proposed, which is of significant importance in improving coordination in MASs. The proposed SV-SCA not only unifies the existing explicit (that is, difference reward) and implicit (that is, VDN and QMIX) SCA mechanisms, but also can be generalized for both MAP and MARL, making it widely applicable to MAS applications. For MAP problems, a coordinated MCTS method is proposed in which SV-SCA is used to decompose the global payoff into individual rewards that can be backpropagated to the root state through the search tree. Guided by SV-SCA, a coordinated MARL is proposed in which each agent performs coordinated learning in a fully decentralized manner. Theoretical analyses show that both SV-SCA based coordinated MCTS and SV-SCA based coordinated MARL can converge. Experimental results validate that 1) for MAP problems, the proposed coordinated MCTS has significant advantages on the solution quality and scalability over independent MCTS and decentralized MCTS methods, and 2) the proposed coordinated MARL based on the SV-SCA exhibits the best performance over existing implicit and explicit SCA-based MARL benchmarks. helps bridge the gap between MAP and MARL in several manners: 1) it allows model-based MAP to help design more efficient model-free MARL methods, and 2) it serves as the basis for deriving principled solutions to fully integrate the two methodologies, where the SCA mechanism is necessary for both MAP and MARL.

References

[1] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proc. TARK*, Aug. 1996, pp. 195–210.

[2] B.-L. Ye et al., "A survey of model predictive control methods for traffic signal control," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 623–640, May 2019.

[3] Y. Li, Y. Zhang, X. Li, and C. Sun, "Regional multi-agent cooperative reinforcement learning for city-level traffic grid signal control," *IEEE/CAA J. Autom. Sinica*, vol. 11, no. 9, pp. 1987–1998, Sep. 2024.

[4] Y. Cui, Z. Xu, L. Zhong, P. Xu, Y. Shen, and Q. Tang, "A task-adaptive deep reinforcement learning framework for dual-arm robot manipulation," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 466–479, 2025.

[5] B. Chen et al., "DIBNN: A dual-improved-BNN based algorithm for multi-robot cooperative area search in complex obstacle environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 2361–2374, 2025.

[6] M. Mishra, P. Poddar, R. Agrawal, J. Chen, P. Tokekar, and P. B. Sujit, "Multi-agent deep reinforcement learning for persistent monitoring with sensing, communication, and localization constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 2831–2843, 2025.

[7] X. Mao, G. Wu, M. Fan, Z. Cao, and W. Pedrycz, "DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 1028–1044, 2025.

[8] S. Ma, J. Ruan, Y. Du, R. Bucknall, and Y. Liu, "An end-to-end deep reinforcement learning based modular task allocation framework for autonomous mobile systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 1519–1533, 2025.

[9] R. Zhao, K. Wang, Y. Li, Y. Fan, F. Gao, and Z. Gao, "Safe multi-agent deep reinforcement learning for the management of autonomous connected vehicles at future intersections," *IEEE Trans. Parallel Distrib. Syst.*, vol. 36, no. 8, pp. 1744–1761, Aug. 2025.

[10] W. Mao, K. Zhang, R. Zhu, D. Simchi-Levi, and T. Başar, "Model-free nonstationary reinforcement learning: Near-optimal regret and applications in multiagent reinforcement learning and inventory control," *Manage. Sci.*, vol. 71, no. 2, pp. 1564–1580, Feb. 2025.

[11] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, Nov. 2002.

[12] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed constraint optimization problems and applications: A survey," *J. Artif. Intell. Res.*, vol. 61, pp. 623–698, Mar. 2018.

[13] Z. Tang et al., "Discovering diverse multi-agent strategic behavior via reward randomization," in *Proc. ICLR*, 2021, pp. 1–14.

[14] X. Lyu, Y. Xiao, B. Daley, and C. Amato, "Contrasting centralized and decentralized critics in multi-agent reinforcement learning," in *Proc. 20th Int. Conf. Auton. Agents MultiAgent Syst. (AAMAS)*, 2021, pp. 844–852.

[15] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.*, 2018, p. 2085.

[16] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. ICML*, May 2018, pp. 4292–4301.

[17] A. K. Agogino and K. Tumer, "Unifying temporal and structural credit assignment problems," in *Proc. 3rd Int. Joint Conf. AutonomousAgents Multiagent Syst. (AAMAS)*, New York, NY, USA, Aug. 2004, pp. 980–987.

[18] A. Agogino and K. Turner, "Multi-agent reward analysis for learning in noisy domains," in *Proc. 4th Int. Joint Conf. Auto. Agents Multiagent Syst.*, Jul. 2005, pp. 81–88.

[19] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI*, vol. 32, 2018, pp. 1397–1404.

[20] D. T. Nguyen, A. Kumar, and H. C. Lau, "Credit assignment for collective multiagent RL with global rewards," in *Proc. NeuIPS*, vol. 31, 2018, pp. 8102–8113.

[21] J. Wang, Y. Zhang, T.-K. Kim, and Y. Gu, "Shapley Q-value: A local reward approach to solve global reward games," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 5, pp. 7285–7292.

[22] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, vol. 1. Cambridge, MA, USA: MIT Press, 1994.

[23] S. S. Fatima, M. Wooldridge, and N. R. Jennings, "A linear approximation method for the Shapley value," *Artif. Intell.*, vol. 172, no. 14, pp. 1673–1699, Sep. 2008.

[24] J. Li et al., "Shapley counterfactual credits for multi-agent reinforcement learning," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 934–942.

[25] S. Han, H. Wang, S. Su, Y. Shi, and F. Miao, "Stable and efficient Shapley value-based reward reallocation for multi-agent reinforcement learning of autonomous vehicles," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 426–433.

[26] Z. Zhou, X. Xu, R. H. L. Sim, C.-S. Foo, and B. K. H. Low, "Probably approximate Shapley fairness with applications in machine learning," in *Proc. AAAI*, vol. 37, 2023, pp. 5910–5918.

[27] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "A unifying framework for reinforcement learning and planning," *Frontiers Artif. Intell.*, vol. 5, pp. 1–25, Jul. 2022.

[28] H. Wei et al., "PressLight: Learning max pressure control to coordinate traffic signals in arterial network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1290–1298.

[29] K. Son, D. W. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. ICML*, 2019, pp. 5887–5896.

[30] J. Wang, Z. Ren, T. Z. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. ICLR*, 2020, pp. 1–11.

[31] S. Liu et al., "Adaptive value decomposition with greedy marginal contribution computation for cooperative multi-agent reinforcement learning," in *Proc. AAMAS*, 2023, pp. 31–39.

[32] D. Han, C. X. Lu, T. Michalak, and M. Wooldridge, "Multiagent model-based credit assignment for continuous control," in *Proc. AAMAS*, 2021, pp. 462–469.

[33] J. Wang, J. Wang, Y. Zhang, Y. Gu, and T. Kim, "SHAQ: Incorporating Shapley value theory into multi-agent Q-learning," in *Proc. NeurIPS*, 2021, pp. 1–14.

[34] C. B. Browne et al., "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[35] L. Kocsis and C. Szepesvári, "Bandit based Monte–Carlo planning," in *Proc. Eur. Conf. Mach. Learn.*, 2006, pp. 282–293.

[36] D. Silver and J. Veness, "Monte–Carlo planning in large POMDPs," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2010, pp. 2164–2172.

[37] T. Hubert, J. Schrittwieser, I. Antonoglou, M. Barekatain, S. Schmitt, and D. Silver, "Learning and planning in complex action spaces," in *Proc. ICML*, 2021, pp. 4476–4486.

[38] A. Lerer, H. Hu, J. Foerster, and N. Brown, "Improving policies via search in cooperative partially observable games," in *Proc. AAAI*, vol. 34, 2020, pp. 7187–7194.

[39] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. P. Bertsekas, "Multiagent rollout and policy iteration for POMDP with application to multi-robot repair problems," in *Proc. CoRL*, 2020, pp. 1814–1828.

[40] D. Claes, F. A. Oliehoek, H. Baier, and K. Tuyls, "Decentralised online planning for multi-robot warehouse commissioning," in *Proc. AAMAS*, vol. 1, 2017, pp. 492–500.

[41] N. Zerbel and L. Yliniemi, "Multiagent Monte Carlo tree search," in *Proc. AAMAS*, 2019, pp. 2309–2311.

[42] A. Czechowski and F. A. Oliehoek, "Decentralized MCTS via learned teammate models," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 81–88.

[43] A. Sivagnanam, A. Pettet, H. Lee, A. Mukhopadhyay, A. Dubey, and Á. Lászka, "Multi-agent reinforcement learning with hierarchical coordination for emergency responder stationing," in *Proc. ICML*, 2024, pp. 1–43.

[44] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 316–337, Mar. 2019.

[45] M. Li, W. Yang, Z. Cai, S. Yang, and J. Wang, "Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty," in *Proc. IJCAI*, 2019, pp. 450–456.

[46] S. Choudhury, J. K. Gupta, P. Morales, and M. J. Kochenderfer, "Scalable anytime planning for multi-agent MDPs," in *Proc. AAMAS*, 2021, pp. 341–349.

[47] C. Amato and F. A. Oliehoek, "Scalable planning and learning for multiagent POMDPs," in *Proc. AAAI*, vol. 29, 2015, pp. 1995–2002.

[48] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis, "Optimal and approximate Q-value functions for decentralized POMDPs," *J. Artif. Intell. Res.*, vol. 32, pp. 289–353, May 2008.

[49] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. NIPS*, 2017, pp. 6379–6390.

[50] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," in *Proc. NeurIPS*, 2020, pp. 1–12.

[51] W. Chen, W. Li, X. Liu, S. Yang, and Y. Gao, "Learning explicit credit assignment for cooperative multi-agent reinforcement learning via polarization policy gradient," in *Proc. AAAI*, vol. 37, 2023, pp. 11542–11550.

[52] B. Peng et al., "FACMAC: Factored multi-agent centralised policy gradients," in *Proc. NeuIPS'21*, 2020, pp. 12208–12221.

[53] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 980–991.

[54] Y. Kang, T. Wang, X. Wu, Q. Yang, and C. Zhang, "Non-linear coordination graphs," in *Proc. NeurIPS*, 2022, pp. 1–12.

[55] S. Li, J. K. Gupta, P. Morales, R. Allen, and M. J. Kochenderfer, "Deep implicit coordination graphs for multi-agent reinforcement learning," in *Proc. AAMAS*, 2020, pp. 764–772.

[56] T. Wang, L. Zeng, W. Dong, Q. Yang, Y. Yu, and C. Zhang, "Context-aware sparse deep coordination graphs," in *Proc. ICLR*, 2021, pp. 1–12.

[57] Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao, "LIIR: Learning individual intrinsic reward in multi-agent reinforcement learning," in *Proc. NeurIPS*, vol. 32, 2019, pp. 4403–4414.

[58] Y. Xiao, X. Lyu, and C. Amato, "Local advantage actor-critic for robust multi-agent deep reinforcement learning," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst. (MRS)*, Nov. 2021, pp. 155–163.

[59] W. Li, B. Jin, X. Wang, J. Yan, and H. Zha, "F2A2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 23, pp. 1–75, Jun. 2020.

[60] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "Adopt: Asynchronous distributed constraint optimization with quality guarantees," *Artif. Intell.*, vol. 161, nos. 1–2, pp. 149–180, Jan. 2005.

[61] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, "Decentralised coordination of low-power embedded devices using the max-sum algorithm," in *Proc. AAMAS*, vol. 2, 2008, pp. 639–646.

[62] M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *Proc. ICML*, 1993, pp. 330–337.

[63] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proc. ICML*, 2002, pp. 227–234.

[64] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *J. Mach. Learn. Res.*, vol. 7, no. 65, pp. 1789–1828, 2006.

[65] C. Zhang and V. Lesser, "Coordinated multi-agent reinforcement learning in networked distributed POMDPs," in *Proc. AAAI*, vol. 25, 2011, pp. 764–770.

[66] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, and C. Zhang, "Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs," in *Proc. AAAI*, 2014, pp. 1447–1455.

[67] H. Zhang et al., "CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario," in *Proc. World Wide Web Conf.*, 2019, pp. 3620–3624.

[68] H. Wei et al., "CoLight: Learning network-level cooperation for traffic signal control," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manag.*, 2019, pp. 1913–1922.

[69] P. Varaiya, "Max pressure control of a network of signalized intersections," in *Proc. Transp. Res. C, Emerg. Technol.*, vol. 36, 2013, pp. 177–195.

[70] H. Santana, G. Ramalho, V. Corruble, and B. Ratitch, "Multi-agent patrolling with reinforcement learning," in *Proc. AAMAS*, 2004, pp. 1122–1129.

[71] A. Rosenfeld, O. Maksimov, and S. Kraus, "When security games hit traffic: A deployed optimal traffic enforcement system," *Artif. Intell.*, vol. 289, Dec. 2020, Art. no. 103381.

[72] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *J. Artif. Intell. Res.*, vol. 19, pp. 399–468, Oct. 2003.

[73] A. K. Agogino and K. Tumer, "Analyzing and visualizing multiagent rewards in dynamic and stochastic domains," *Auto. Agents Multi-Agent Syst.*, vol. 17, no. 2, pp. 320–338, Oct. 2008.
[74] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. ICML*, 2000, pp. 535–542.
[75] E. Marchesini, A. Baisero, R. Bhati, and C. Amato, "On stateful value fctorization in multi-agent reinforcement learning," in *Proc. AAMAS*, 2025, pp. 1445–1453.

Wanyuan Wang (Member, IEEE) received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2016. He is currently an Associate Professor with the School of Computer Science and Engineering, Southeast University. He has published over 50 papers in refereed journals and conference proceedings, such as IEEE TRANSACTIONS, NeurIPS, AAAI, and AAMAS. His research interests include artificial intelligence, multiagent systems, LLM agents, and embodied intelligence. He won the Best Student Paper Award from ICTAI14.

Qian Che received the Ph.D. degree in cyber science and engineering from Southeast University, China, in 2024. She is currently a Lecturer with Jiangsu Police Institute. Her research interests include multiagent systems.

Chunyu Liu is currently pursuing the M.E. degree with the School of Computer Science and Engineering, Southeast University. His research interests include multiagent reinforcement learning.

Youzhi Zhang received the Ph.D. degree in computer science from Nanyang Technological University, Singapore. He is currently an Assistant Professor with the Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences, Hong Kong. His research interests include AI, multiagent systems, and computational game theory.

Jiuchuan Jiang received the Ph.D. degree in computer science from Nanyang Technological University in 2019. He is currently an Associate Professor with the School of Information Engineering, Nanjing University of Finance and Economics. He has published over 50 papers at premier conferences and journals, including KDD, AAMAS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, ACM TIST, and ACM TAAS. His research interests include artificial intelligence, multiagent systems, and social networks. He has won the 2020 Wu Wenjun Artificial Intelligence Award and the 2020 Jiangsu Provincial Science and Technology Award. He serves as the Area Chair for top conferences, such as ICML and NeurIPS, and he also regularly serves as a program committee member/a reviewer for premier AI conferences, such as AAAI, IJCAI, AAMAS, ICLR, and ECAI.

Bo An (Senior Member, IEEE) is currently the President's Council Chair Associate Professor of Computer Science and Engineering and the Co-Director of Artificial Intelligence Research Institute (AIR), Nanyang Technological University, Singapore. He has published over 100 referred papers at AAMAS, IJCAI, AAAI, ICAPS, KDD, UAI, EC, WWW, ICLR, NeurIPS, ICML, JAAMAS, AIJ, and ACM/IEEE TRANSACTIONS. His current research interests include artificial intelligence, multiagent systems, computational game theory, reinforcement learning, and optimization. He was named to IEEE Intelligent Systems' "AI's 10 to Watch" list for 2018. He was invited to be an Advisory Committee Member of IJCAI'18. He is the PC Co-Chair of AAMAS'20. He is a member of the Editorial Board of JAIR and is an Associate Editor of AIJ, JAAMAS, IEEE INTELLIGENT SYSTEMS, ACM TAAS, and ACMTIST. He was elected to the board of directors of IFAAMAS, a Senior Member of AAAI, and a Distinguished Member of ACM.