Structural Credit Assignment-Guided Coordinated MCTS: An Efficient and Scalable Method for Online Multiagent Planning

Qian Che
School of Cyber Science and Engineering, Southeast University
Nanjing, China
qche@seu.edu.cn

Wanyuan Wang*
School of Computer Science and Engineering, Southeast University
Nanjing, China
wywang@seu.edu.cn

Fengchen Wang
School of Computer Science and Engineering, Southeast University
Nanjing, China
fcwang@seu.edu.cn

Tianchi Qiao
School of Computer Science and Engineering, Southeast University
Nanjing, China
tianchi-qiao@seu.edu.cn

Xiang Liu
School of Computer Science and Engineering, Southeast University
Nanjing, China
xiangliu@seu.edu.cn

Jiuchuan Jiang
Nanjing University of Finance and Economics
Nanjing, China
jcjiang@nufe.edu.cn

Bo An
School of Computer Science and Engineering
Nanyang Technological University
Singapore
boan@ntu.edu.sg

Yichuan Jiang*
School of Cyber Science and Engineering, Southeast University
Nanjing, China
yjiang@seu.edu.cn

ABSTRACT

Online planning has been widely focused in many areas, such as industry chain and collective intelligence. Due to the trade-off nature of trading computation time for solution quality, Monte-Carlo tree search (MCTS) methods have shown great success in online planning. However, the exponential growth of global joint-action space makes it challenging to apply MCTS to online multiagent planning (MAP). Our goal in this paper is to design an efficient and scalable coordinated MCTS method for online MAP. Combining with coordination graphs, recent Factored Value MCTS (FV-MCTS) has attempted to recover the trade-off property for MCTS-based online MAP. However, FV-MCTS directly uses the global payoff to reward each agent, and has difficulty in finding coordination actions in multiagent MCTS settings where other agents are also taking exploratory actions. We overcome this limitation by designing a generalized structural credit assignment (SCA)-guided coordinated MCTS, where SCA is used to promote coordination and MCTS is used to search promising global joint-actions. Specially, we use the Shapley value to provide a fair SCA, which can be efficiently computed by exploiting locality of interaction between agents. Moreover, theoretical analysis shows that the proposed method can bound the bias of the estimated value of the global join-action under certain conditions. Finally, we conduct extensive experiments in some typical sequential multiagent coordination domains such as multi-robot warehouse patrolling in industry chain, etc. to validate the efficiency and scalability of the proposed method over other benchmarks.

KEYWORDS

Collective Intelligence; Multiagent Planning; Coordinated MCTS; Credit Assignment; Shapley Value

ACM Reference Format:
Qian Che, Wanyuan Wang*, Fengchen Wang, Tianchi Qiao, Xiang Liu, Jiuchuan Jiang, Bo An, and Yichuan Jiang*. 2023. Structural Credit Assignment-Guided Coordinated MCTS: An Efficient and Scalable Method for Online Multiagent Planning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

With the rapid development of industrial production, multi-robot systems have a wide range of application in the industrial chain [25] and collective intelligence [17, 38]. For example, in multi-robot warehouse patrolling, a team of mobile robots are deployed to patrol along regions for implementing persistent surveillance, inspection and control in industry chain [10, 11]. In such a cooperative multiagent system (MAS), coordinating these agents (e.g., robots) that take sequences of actions to optimize the long-term global payoff is of crucial importance [7]. Sequential multiagent coordination problems can be formulated as a multiagent Markov decision process (MMDP), which has been shown to be computationally intractable in general [4]. Efficient approximation methods for MMDPs have attempted to alleviate the computational challenge and trade off optimality for scalability. *Offline* multiagent reinforcement learning (MARL) and *online* multiagent planning (MAP) are two important classes of approximation methods.

State-of-the-art offline MARL methods have widely adopted the framework of centralized training for decentralized execution, where the policy function of each agent is trained offline using centralized information and query the policy during execution

*corresponding author.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

in a decentralized manner [28, 34]. However, such decentralized MARL always converge to a non-cooperative equilibrium with non-optimal payoffs due to the randomized policies of agents [26, 35]. Online MAP methods use a quite different strategy to deal with the large joint-action spaces of MMDPs: they focus only on states that are reachable from the current state, while computing the next action at each online planning step [13, 27, 41]. Due to the trade-off between computation time and solution quality [20], Monte-Carlo tree search (MCTS) has been recognized as one of the most powerful online planning methods [31–33]. Unfortunately, because there are exponential number of global joint-actions required to be evaluated, naive application of MCTS (i.e., search global joint-actions) for online MAP negates the trade-off nature of MCTS.

The key motivation of this paper is to design an efficient and scalable coordinated MCTS method for online MAP. Recently, combining with coordination graphs, Factored Value MCTS (FV-MCTS) [3, 9] has attempted to recover the trade-off property of MCTS-based online MAP. To avoid exploring global joint-actions, FV-MCTS allows each agent to explore individual actions directly by using the global payoff. Since each agent's reward depends on the actions of all the other agents, the global payoff-based reward mechanism might fail to find coordination behaviors in non-stationary multiagent MCTS settings where other agents are also taking exploration actions [2]. Structural credit assignment (SCA), which determines how a single agent's action contributes to global payoff, is critical for evaluating an agent's action for MASs [1]. Our goal in this paper is designing SCA-guided coordinated MCTS, where SCA is used to promote coordination and MCTS is used to select the most promising coordinated joint-action that can maximize the long-term global payoffs in an anytime manner.

In summary, the first contribution of this paper is to propose a generalized SCA-guided coordinated MCTS method for online MAP, which unifies existing multiagent MCTS methods including Decentralized MCTS with difference credit [42] and FV-MCTS with global payoff credit [3, 9]. Drawing inspiration from cooperative game theory, a Shapley value-based SCA is proposed to provide a fair decomposition of the global payoff to each agent. However, computing the Shapley value is often #P-complete [15]. Exploiting locality of interaction between agents, our second technical contribution is efficiently computing the Shapley value. By carefully mapping the state-action value of each agent to local joint-action value, our third contribution is to bound the bias of estimated value of the global joint-action under certain conditions. Finally, we conduct extensive experiments in some domains such as multi-robot warehouse patrolling in industry chain, etc. to validate the efficiency and scalability of the proposed SCA-guided coordinated MCTS method over existing offline MARL and online MAP benchmarks.

2 RELATED WORK

MCTS-based online MAP. MCTS is a well-known online planning strategy for constructing anytime solutions for sequential decision making [8]. Since the number of global joint-actions grow exponentially, naive applications of MCTS for online MAP has difficulty in trading-off exploration and exploitation. To reduce search complexity, single-agent MCTS simply allows one specific agent to search the policy while all other agents play a fixed common-knowledge policy [22]. Decentralized MCTS (Dec-MCTS) allows all agents to search their own policies in turn [6]. In Dec-MCTS, the behaviors of other agents can be modeled by greedy heuristics [10, 42] or learned from previous experience [12]. To improve modeling accuracy, agents can also communicate and share their policies with other agents [5, 24]. However, these independent MCTS methods might yield suboptimal solutions in complex multiagent problems where reasoning about the effect of joint-actions is necessary [9]. Our proposed coordinated MCTS attempts to explore and evaluate the joint-action effectively for better solution quality.

In many realistic MASs, agents interact only with a subset of local agents, where coordination graphs (CGs) can be used to model such local interactions [16]. Recently, combining with CGs, Factored Value MCTS (FV-MCTS) attempts to explore local joint-actions according to the Upper Confidence Bound (UCB) statistics [3, 9]. Existing FV-MCTS directly uses the global payoff (which can be computed exactly by Variable Elimination [3] or approximately by Max-sum [9]) to reward each agent. This individual reward is used to evaluate the local joint-action value. Since each agent's reward depends on actions of all the other agents, the global payoff-based reward mechanism generally suffers from low-payoff non-cooperative solutions in dynamic mutiagent environments with a number of exploratory agents [2]. To address the limitation imposed by FV-MCTS, this paper designs an SCA-guided coordinated MCTS, where SCA can help to find high-payoff coordinated solutions.

Structural Credit assignment (SCA) for MARL. SCA aims to distribute the global payoff to each agent's individual payoff, and has been a central research topic in MARL. Prior SCA methods for MARL can be divided into the implicit and explicit approaches. In implicit methods, the relations between individual and global Q-values are learned through neural networks [28, 34]. However, such implicit global payoff decomposition functions make the joint policy converge to suboptimal solutions [23]. With static coordination structures, the state-action value of the local interaction function can be learned by the standard Q-learning and the global joint-action can be computed by the Max-sum-based coordination algorithm [18, 21]. With dynamic coordination structures, Agogino and Tumer [1, 2] first propose an explicit method, where each agent is rewarded by the difference between global payoff and the counterfactual global payoff without this agent. Recently, borrowing the concept from the cooperative game-theory [15], the Shapley value has been proposed as an alternative explicit method, where each agent is rewarded by the average marginal contribution to each subset of other agents [23, 37]. Since computing the exact Shapley value is intractable, recent studies usually use an approximation of the Shapley value as a substitution. As the main technical contribution of this paper, by exploiting the local coordination structure between agents, we propose a computationally efficient algorithm to compute the exact Shapley value.

3 PROBLEM DESCRIPTION

In this section, we start by introducing the multiagent Markov decision process (MMDP) model which is suitable for various sequential multiagent coordination problems. We then briefly describe the framework of structural credit assignment (SCA)-guided coordinated MCTS for MMDPs.

MMDP Model. Formally, an MMDP can be defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$: $\mathcal{N} = \{1, 2, \cdots, n\}$ is the set of n agents, $\mathcal{S}$ is a finite set of global states s of the environment. In an MMDP, each agent can observe the global state. $\mathcal{A} = A_1 \times \cdots \times A_n$ is 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 up 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 the joint action $\vec{a}$ at state s and ending at state s'.

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

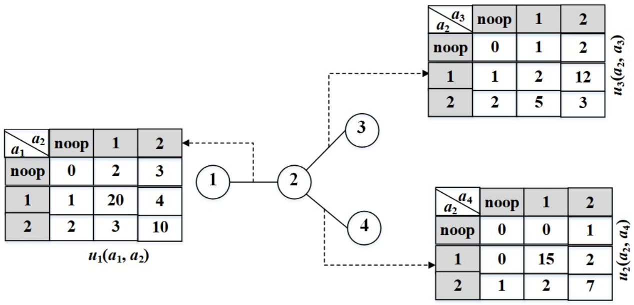
The Framework of SCA-guided Coordinated MCTS. The proposed SCA-guided coordinated MCTS framework mainly comprises two parts:

- **Part 1: Shapley Value-based SCA for Individual Action Evaluation**. In this part, we assume a stateless or single-state multiagent coordination setting. The goal of this part is to find the optimal joint-action that maximizes the global payoff and decompose the optimal global payoff for evaluating the action of each agent. Such individual action evaluation will be used for individual agent selection and global payoff backpropagation in the coordinated MCTS part. The main contribution of this part is to use the Shapley value to provide a fair individual action evaluation and propose a computationally efficient algorithm to compute the exact Shapley value.
- **Part 2: SCA-Guided Coordinated MCTS**. In this part, we apply MCTS for exploring the most promising joint-action that maximizes the cumulative long-term global payoff. The main contribution of this part is to employ SCA to decompose the state-independent global payoff to reward each agent and backpropagate the agent reward for joint-action evaluation.

4 SHAPLEY VALUE-BASED SCA FOR INDIVIDUAL ACTION EVALUATION

This section first reviews a popular multiagent coordination model that represents local interactions between agents. We then describe Shapley value-based SCA and show how to compute the Shapley value efficiently by exploiting the multiagent coordination model.

Distributed Constraint Optimization (DCOP). Many multiagent coordination problems demonstrate the *locality of interaction*, i.e., an agent's action only has an impact on the actions of a subset of locally interacted agents. DCOP has emerged as a key formalism for such settings where the primary interactions are between local subsets of agents [16]. 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 $D_i \in \mathcal{D}$, 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, i.e., $u_j : D_{j_1} \times D_{j_2} \times \cdots \times D_{j_k} \longrightarrow \mathbb{R}_{\geq 0}$. Let $\mathcal{N}_{u_j}$ denote the set of agents involved in the utility u_j. Without loss of generality, let *noop* denote the no operation action, and $\forall u_j$, $u_j(\vec{noop}) = 0$, i.e., there will be no utilities 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, i.e., $\vec{a}^* = \arg\max_{\vec{a}} u(\mathcal{N})$.



a_1 \ a_2	noop	1	2
noop	0	2	3
1	1	20	4
2	2	3	10

$u_1(a_1, a_2)$

a_2 \ a_3	noop	1	2
noop	0	1	2
1	1	2	12
2	2	5	3

$u_3(a_2, a_3)$

a_2 \ a_4	noop	1	2
noop	0	0	1
1	0	15	2
2	1	2	7

$u_2(a_2, a_4)$

Figure 1: A toy DCOP problem, and the optimal global joint-action $\vec{a} = \{1, 1, 2, 1\}$.

Exploiting the structure of locality of interaction, we can find that an agent i's action has an effect only on the utility functions that involves i. We give a formal definition of *local interacted agents*, which is useful for computing the Shapley value.

Definition 1. ***Local Interacted Agents.*** *Given a DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$, an agent k is defined as the local interacted agent of agent i if there exists one utility function that involves both i and k, i.e., the local interacted agents of i, $\mathcal{N}_i^{loc} = \{k | \exists u_j \in \mathcal{U} : i \in \mathcal{N}_{u_j} \& k \in \mathcal{N}_{u_j}\}$.*

Example 1. *Figure 1 shows a DCOP consisting of four agents $\{1, 2, 3, 4\}$. Each agent can select the actions of* noop *, 1 and 2. This DCOP also comprises three utility functions $u_1(a_1, a_2)$, $u_2(a_2, a_4)$ and $u_3(a_2, a_3)$. Agent 1 only interacts with agent 2, thus $\mathcal{N}_1^{loc} = \{2\}$. Similarity, we have $\mathcal{N}_2^{loc} = \{1, 3, 4\}$,$\mathcal{N}_3^{loc} = \{2\}$, and $\mathcal{N}_4^{loc} = \{2\}$.*

Max-sum for Optimizing DCOP. Given a DCOP, Max-sum algorithm has been proposed for optimizing the global joint action $\vec{a}^*$ in a fully decentralized manner [14]. The pseudocode of standard Max-sum is shown in Algorithm 1. Max-sum first transforms the DCOP to 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 a variable node i with a function node u_j if and only if i is involved in u_j. Variable nodes and function nodes can perform computation and send message. On the factor graph, let Neg_i (resp. Neg_{u_j}) denote the set of neighbor function nodes (resp. variable nodes) of the variable node i (resp. function node u_j). The operations for variable and function nodes are similar apart from the content of messages to be sent. A message sent from a variable node i to a function node u_j at iteration k, includes for each value $a_i \in D_i$ the sum of utilities for this value it received from all function neighbors apart from u_j at iteration $k - 1$. Formally, the message from variable node i to function node u_j includes for each value $a_i \in D_i$:

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

Algorithm 1: Max-sum (0)

```
Input  : The DOCP $\langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$.
Output : Joint action $\vec{a}^*$.
for $k$ iterations do
    for variable node $i$ do
        produce message $E_{i \to u_j}$ using messages from
          $Neg_i \setminus \{u_j\}$;
    for funtion node $u_j$ do
        produce message $F_{u_j \to i}$ using messages from
          $Neg_{u_j} \setminus \{i\}$;
Compute the joint action $\vec{a}^*$ by Eq.(3);
```

where $F_{u_{j'} \to i}(a_i)$ is the utility for value a_i included in the messages received from the function node $u_{j'}$ at iteration $k-1$. α is a constant to prevent utilities carried by messages from growing arbitrarily.

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

$$F_{u_j \to i}(a_i) = \max_{\vec{a}_{u_j} \setminus a_i} \big(u_j(\vec{a}_{u_j}) + \textstyle\sum_{i' \in Neg_{u_j}, i' \neq i} E_{i' \to u_j}(a_{i'})\big). \quad (2)$$

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

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

Max-sum converges to the optimum for DCOP with acyclic factor graphs and provides desirable solutions in cyclic factor graphs [14].

4.1 Shapley Value-based SCA

Given a DCOP, after Max-sum computes the optimal global joint-action $\vec{a}^*$ that maximizes the global payoff, the SCA problem is to evaluate how the individual action a_i^* of agent i contributes to the global payoff, and is critical for promoting coordination. In this section, we describe the core idea of our Shapley value-based SCA and show how to efficiently compute the Shapley value by exploiting the structure of the DCOP.

The Shapley value has recently been used for SCA in multiagent RL tasks [19, 23, 37]. Since computing the exact Shapley value is often intractable [15], previous methods use an approximation of Shapley value as a substitution. Our contribution is to exploit the locality of interactions of the DCOP, and design a computationally efficient algorithm to compute the exact Shapley value.

Marginal Contribution. Given a DCOP $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U} \rangle$ and a global joint-action $\vec{a} = \{a_1, \cdots, a_n\}$ returned by Max-sum, let $u(C) = \sum_{u_j} u_j(a_{j_1}^C, \cdots, a_{j_k}^C)$ denote the total utility achieved by the coalition $C \subseteq \mathcal{N}$, where $a_i^C = a_i$ if the agent $i \in C$, $a_i^C = noop$ otherwise. The marginal contribution $\triangle_i^{\vec{a}}$ of the agent i to a coalition $C \subseteq \mathcal{N} \setminus \{i\}$ is the increase in the utility of C as a result of i joining it and taking action a_i, i.e.,

$$\triangle_i^{\vec{a}}(C) = u(C \cup \{i\}) - u(C). \quad (4)$$

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

$$\phi_i^{\vec{a}} = \textstyle\sum_{C \subseteq \mathcal{N} \setminus \{i\}} \frac{|C|!(n-|C|-1)!}{n!} \triangle_i^{\vec{a}}(C). \quad (5)$$

The Shapley value can be interpreted as that all agents are arranged in some order, all orderings being equally likely, and then $\phi_i^{\vec{a}}$ is the expected marginal contribution, over all orderings, of agent i to the set of agents who precede him.

Shapley value-based SCA is to assign the credit of agent i by the Shapley value $\phi_i^{\vec{a}}$. This value not only gives a fair division of the utilities of coordination between agents, but also satisfies the *efficiency* property [15].

Lemma 1. ***Efficiency.*** *Given 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, i.e., $\sum_i \phi_i^{\vec{a}} = u(\mathcal{N})$.*

Computing the Shapley value. Although Shapley value provides a fair SCA, its main drawback is the complexity of computing it. Fortunately, in DCOP, an agent i's action has an effect only on the utilities of the coalition that involves the locally interacted agents of i. This insight indicates that the marginal contribution of i to all coalitions that do not involve any locally interacted agent of i is the same (i.e., zero). We illustrate this insight using Example 1 again.

Example 1 (cont.). *In Figure 1, the optimal global joint-action is $\vec{a} = \{1, 1, 2, 1\}$, where $u_1(a_1, a_2) = 20$, $u_2(a_2, a_4) = 15$, $u_3(a_2, a_3) = 12$. Agent 1 only interacts with agent 2, then, agent 1's action does not have an effect on the utilities of these irrelevant coalitions $C_1 = \{3\}$, $C_2 = \{4\}$, and $C_3 = \{3, 4\}$. The marginal contribution of the agent 1 to all these irrelevant coalitions will be zero, i.e., $\triangle_1^{\vec{a}}(C_i) = 0$ $(1 \leq i \leq 3)$. On the other hand, for these relevant coalitions $C_4 = \{2\}$, $C_5 = \{2, 3\}$, $C_6 = \{2, 4\}$, and $C_7 = \{2, 3, 4\}$ that involve the locally interacted agent 2, agent 1's marginal contributions to them are also the same, i.e., $\triangle_1^{\vec{a}}(C_i) = u(C_i \cup \{i\}) - u(C_i) = 18$ $(4 \leq i \leq 7)$.*

The above example indicates that it is typically not necessary to enumerate all possible coalitions, and finding these common coalitions that have the same marginal contribution can reduce the computation complexity. In the following, we exploit such interaction structure and compute the exact Shapley value efficiently.

Theorem 1. *Given the agent i, let $\mathcal{N}_i^{loc}$ denote the subset of the locally interacted 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 agent i and $n_i^{irr} = |\mathcal{N}_i^{irr}|$. The Shapley value of i, $\phi_i^{\vec{a}}$ then can be rewritten by*

$$\phi_i^{\vec{a}} = \frac{1}{n!} \textstyle\sum_{C \subseteq \mathcal{N}_i^{loc}} H(|C|, n_i^{irr}) \triangle_i^{\vec{a}}(C). \quad (6)$$

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

$$H(|C|, n_i^{irr}) = \textstyle\sum_{k=0}^{n_i^{irr}} \binom{n_i^{irr}}{k} (|C| + k)!(n - |C| - k - 1)! \quad (7)$$

Proof. For each subset of agents C, dividing it into disjoint coalitions of locally interacted agents $C_1 \subseteq \mathcal{N}_i^{loc}$ and irrelevant agents $C_2 \subseteq \mathcal{N}_i^{irr}$ of agent i, we have

$$
\begin{aligned}
\phi_i^{\vec{a}} &= \frac{1}{n!}\sum_{C\subseteq N\setminus\{i\}}|C|!(n-|C|-1)!\triangle_i^{\vec{a}}(C)\\
&= \frac{1}{n!}\sum_{C_1\subseteq \mathcal{N}_i^{loc}}\sum_{C_2\subseteq \mathcal{N}_i^{irr}}|C_1\cup C_2|!(n-|C_1\cup C_2|-1)!\triangle_i^{\vec{a}}(C_1\cup C_2)\\
&= \frac{1}{n!}\sum_{C_1\subseteq \mathcal{N}_i^{loc}}\sum_{C_2\subseteq \mathcal{N}_i^{irr}}|C_1\cup C_2|!(n-|C_1\cup C_2|-1)!\triangle_i^{\vec{a}}(C_1) \qquad (8)\\
&= \frac{1}{n!}\sum_{C_1\subseteq \mathcal{N}_i^{loc}}\sum_{C_2\subseteq \mathcal{N}_i^{irr}}(|C_1|+|C_2|)!(n-|C_1|-|C_2|-1)!\triangle_i^{\vec{a}}(C_1)\\
&= \frac{1}{n!}\sum_{C_1\subseteq \mathcal{N}_i^{loc}}\sum_{k=0}^{n_i^{irr}}\binom{n_i^{irr}}{k}(|C_1|+k)!(n-|C_1|-k-1)!\triangle_i^{\vec{a}}(C_1)\\
&= \frac{1}{n!}\sum_{C_1\subseteq \mathcal{N}_i^{loc}}H(|C_1|,n_i^{irr})\triangle_i^{\vec{a}}(C_1).
\end{aligned}
$$

The reason that Eq.(8) holds is that given a local coalition $C_1 \subseteq \mathcal{N}_i^{loc}$, for any irrelevant coalition $C_2 \subseteq \mathcal{N}_i^{irr}$, the agent i has the same marginal contribution to such a joint coalition $C_1 \cup C_2$, that is $\forall C_2, C_2' \subseteq \mathcal{N}_i^{irr}$, $\triangle_i^{\vec{a}}(C_1 \cup C_2) = \triangle_i^{\vec{a}}(C_1 \cup C_2')$. □

It should be noted that the operator $H(|C|, n_i^{irr})$ only depends on the *size* of local coalitions (i.e., $|C|$) and the *number* of these irrelevant agents n_i^{irr}, rather than the actions of other agents. Thus, we can pre-compute and store $H(x, y)$ $(1 \le y \le n, 0 \le x \le n-y)$ in an offline manner to mitigate the online computation load.

5 SCA-GUIDED COORDINATED MCTS

Combining Shapley value-based SCA, this section presents coordinated MCTS for MMDPs. Since search is time consuming and the coordination structure is dynamic (i.e., the DCOP is state-dependent), the key implementation procedures of coordinated MCTS are individual action selection and global payoff backpropagation.

Individual Action Selection. Given the current state s, we first formulate the current DCOP $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U}\rangle$. On the one hand, agents need to coordinate their joint-action $\vec{a}$ by Max-sum and optimize the immediate global payoff. On the other hand, it is also necessary to explore promising joint-actions that are beneficial for the long-term cumulative global payoffs. As mentioned previously, directly exploring the global joint-action is infeasible. By exploiting the message-passing procedure in a DCOP, we can instead explore the individual action a_i of agent i at the final action selection procedure (i.e., Eq.(3)). Specifically, we keep track of corresponding frequency statistics for each individual action a_i of agent i. A natural exploration over individual action is to add the Upper Confidence Bound (UCB) bonus during the action selection:

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

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

Global Payoff Backpropagation. In the standard single-agent MCTS, the values of all ancestor nodes of the root node s are updated by backpropagating the payoff from the leaf nodes to s. However, in MMDPs, exploring the global joint-actions is infeasible, and directly backpropagating the global payoff to update global joint-action statistics can not apply. We address this issue by employing Shapley value-based SCA to decompose the global payoff to individual reward, and backpropagating the individual reward to update the statistics of individual actions as well as local joint-actions.

Algorithm 2: SCA-guided Coordinated MCTS

```
Input  : time limit, depth, $c$, state $s$, discounted factor $\gamma$.
Output : Joint Action $\vec{a}^*$.
while time limit not reached do
    SIMULATE($s$,depth);
$\vec{a}^* \leftarrow$ Max-sum(0) by Algorithm 1;
Function Simulate($s$,DEPTH):
    if DEPTH=0 then
        return 0;
    INITIALIZESTATE($s$);
    $\vec{a} \leftarrow$ MAX-SUM(c), $s' \sim P(s, \vec{a}, s')$;
    for each agent $i \in \mathcal{N}$ do
        $R_i(s, a_i, s') = \phi_i^{\vec{a}}$ via Eq.(6);
        $q_i(s, a_i) \leftarrow$
          $R_i(s, a_i, s') + \gamma \cdot$ SIMULATE($s'$, depth $- 1$).$R_i$;
    UPDATESTATE($s$,$\vec{a}$,$\overline{u}$);
Function InitializeState($s$):
    if $s$ is a new state then
        Formulate $\mathcal{G}(s) = \langle \mathcal{N}, \mathcal{A}, \mathcal{D}, \mathcal{U}\rangle$;
        for each agent $i \in \mathcal{N}$ & $a_i \in D_i$ do
            Initialize $N_i(s, a_i) = 0$;
        for each utility function $u_j \in \mathcal{U}$ & $\vec{a}_{u_j} \in D_{u_j}$ do
            Initialize $\overline{u}_j(s, \vec{a}_{u_j}) = 0$, and $N_{u_j}(s, \vec{a}_{u_j}) = 0$;
Function Max-Sum($c$):
    Steps 1-5 of Max-sum(0) by Algorithm 1;
    Compute the global joint-action $\vec{a}^*$ by Eq.(9);
Function Updatestate($s$,$\vec{a}$,$\overline{u}$):
    for each agent $i \in \mathcal{N}$ do
        $N_i(s, a_i) + = 1$;
    for each utility function $u_j \in \mathcal{U}$ do
        $N_{u_j}(s, \vec{a}_{u_j})$+=1, $q_{u_j}(s, \vec{a}_{u_j}) = \sum_{i\in Neg_{u_j}(s)} \frac{q_i(s,a_i)}{|Neg_i(s)|}$;
        $\overline{u}_j(s, \vec{a}_{u_j}) + = \frac{q_{u_j}(s,\vec{a}_{u_j}) - \overline{u}_j(s,\vec{a}_{u_j})}{N_{u_j}(s,\vec{a}_{u_j})}$;
```

Updating Individual Action and Local Utility Function. Given the 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 by Eq.(9), and $R(s, \vec{a}, s')$ denote the immediate global reward achieved by taking $\vec{a}$ and ending the state s'. We first use the Shaley value to determine the immediate individual reward, i.e., $\forall i, R_i(s, a_i, s') = \phi_i^{\vec{a}}$. In coordinated MCTS, the key is to model each local utility function $u_j \in \mathcal{U}$ at state s, which is necessary for individual action selection. Let $q_i(s, a_i)$ denote the discounted cumulative individual reward of i by taking action a_i at state s. The discounted cumulative rewards of each local utility function u_j can be defined as the sum of the weighted cumulative individual rewards of agent $i \in Neg_{u_j}(s)$, i.e.,

$$q_{u_j}(s, \vec{a}_{u_j}) = \sum_{i\in Neg_{u_j}(s)} \frac{q_i(s,a_i)}{|Neg_i(s)|}. \qquad (10)$$

where $Neg_{u_j}(s)$ denotes the set of neighbor agents of u_j and $Neg_i(s)$ denotes the neighbors of the agent i at current DCOP $\mathcal{G}(s)$. This way of updating local utility function can preserve the efficiency property of 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 any global joint-action, $\phi_i^{\vec{a}}$ denote the Shapley value of agent i, and $q_{u_j}(\vec{a}_{u_j}) = \sum_{i \in Neg_{u_j}} \frac{\phi_i^{\vec{a}}}{|Neg_i|}$ denote the decomposed value of local utility function u_j. By summing the values of all utility functions, we also guarantee the efficiency property, i.e., $u(\mathcal{N}) = \sum_{u_j \in \mathcal{U}} q_{u_j}(\vec{a}_{u_j})$.*

Finally, we define $\overline{u}_j(s, \vec{a}_{u_j})$, the mean of the local utility function u_j obtained so far when the local joint-action $\vec{a}_{u_j}$ is selected at state s, and update it standard average of the simulated future reward:

$$\overline{u}_j(s, \vec{a}_{u_j}) = \overline{u}_j(s, \vec{a}_{u_j}) + \frac{q_{u_j}(s, \vec{a}_{u_j}) - \overline{u}_j(s, \vec{a}_{u_j})}{N_{u_j}(s, \vec{a}_{u_j})}. \tag{11}$$

where $N_{u_j}(s, \vec{a}_{u_j})$ is the visit frequency of the local joint-action $\vec{a}_{u_j}$. The local utility statistics $\overline{u}_j(s, \vec{a}_{u_j})$ represent the cumulative long-term utility function, which will be used in message passing procedure (i.e., Eq.(2)) for the global joint-action selection.

5.1 The Algorithm

The SCA-based coordinated MCTS algorithm is formally shown in Algorithm 2, which cycles among 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 individual action $N_i(s, a_i)$ and local joint-action $N_{u_j}(s, \vec{a}_{u_j})$, and the mean value of the utility $\overline{u}_j(s, \vec{a}_{u_j})$ for each local joint action $\vec{a}_{u_j}$ (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 Shapley value to determine the individual reward for each agent i , i.e., $R_i(s, a_i, s') = \phi_i^{\vec{a}}$ (Step 10). At Step 11, by further simulation, the cumulative state-action reward $q_i(s, a_i)$ of agent i is computed by summing the discounted future individual reward $\gamma \cdot$ Simulate(s', depth $-$ 1).R_i, where .R_i indicates the individual reward of agent i returned at state s'.

Max-sum(c) (Steps 20-22): once the message passing procedure (i.e., Steps 1-5 in Algorithm 1, Max-sum(0)) converges, each agent selects his individual action by Eq.(9), in which the parameter c makes a tradeoff between exploration and exploitation.

Updatestate(s,$\vec{a}$,$\vec{u}$) (Steps 23-28): at state s, given the global joint-action $\vec{a}$, for each utility function u_j, the cumulative local joint-action reward $q_{u_j}(s, \vec{a}_{u_j})$ is computed by summing the weighted individual rewards of neighbor agents $i \in Neg_{u_j}$ (i.e., Eq.(10)). The mean statistic of the local utility function u_j with respect to the local joint-action $\vec{a}_{u_j}$, $\overline{u}_j(s, \vec{a}_{u_j})$ then is updated by Eq.(11).

5.2 Theoretical Analysis

We first show that state-independent global payoffs can be backprobagated to the root state by our coordinated MCTS.

Lemma 3. *Let $\rho = \langle s_0, s_1, \cdots, s_T \rangle$ denote the trace of states visited by the simulated global joint-action $\langle \vec{a}_0, \vec{a}_1, \cdots, \vec{a}_{depth} \rangle$. The cumulative global payoff of ρ is $V^\tau(s_0, \vec{a}_0) = \sum_{k=0}^{depth} \gamma^k R(s_k, \vec{a}_k, s_{k+1})$, where $R(s_k, \vec{a}, s_{k+1})$ is the global payoff of the global joint-action $\vec{a}$ at state s_k. At the root state s_0, let $q_{u_j}(s_0, \vec{a}_{0,u_j}, s_1)$ denote the cumulative local utilities of u_j obtained from ρ, we have*

$$V^\tau(s_0, \vec{a}_0) = \textstyle\sum_{u_j \in \mathcal{G}(s_0)} q_{u_j}(s_0, \vec{a}_{0,u_j}, s_1) \tag{12}$$

Our main result is to bound the bias between the global payoff achieved by our coordinated MCTS and the optimal global payoff at the root state under certain conditions.

Theorem 2. *At the current state s, let $u^*(\mathcal{N})$ denote the optimal expected global payoff and $\mathbb{E}[\overline{u}(\mathcal{N})]$ denote the expected global payoff returned by coordinated MCTS. Under the condition that except the specific utility function $u_{j'}$, the bias between the expected value of other local utility functions $\overline{u}_j$ and its real value is bounded by ϵ, i.e., $\forall u_j \neq u_{j'}, \vec{a}_{u_j} \in D_{u_j}, |\mathbb{E}[\overline{u}_j(\vec{a}_{u_j})] - u_j^*(\vec{a}_{u_j})| \leq \epsilon$, the bias between $u^*(\mathcal{N})$ and $\mathbb{E}[\overline{u}(\mathcal{N})]$, $|\mathbb{E}[\overline{u}(\mathcal{N})] - u^*(\mathcal{N})| = O(\frac{lnT}{T})$, where T is the number of simulations at state s.*

Proof. For single-agent MCTS, let $\mathbb{E}[\overline{Q}_t(s, a)]$ denote the estimated payoff of taking action a at state s, and $Q^*(s, a)$ be the optimal (expected) payoff that can be returned by the action a, then the bias between $Q^*(s, a)$ and $\mathbb{E}[\overline{Q}_t(s, a)]$ can be bounded $O(\frac{lnT}{T})$. To achieve this bound, two sufficient conditions should be satisfied [20]:

Condition 1: The expected payoff $\overline{Q}_t(s, a)$ should be updated by the average of the simulated payoff, i.e.,

$$\overline{Q}_t(s, a) = \overline{Q}_{t-1}(s, a) + \frac{v(s, a) - \overline{Q}_{t-1}(s, a)}{N(s, a)} \tag{13}$$

where $v(s, a)$ is the discounted cumulative payoff returned by taking a at state s at the $(t-1)$th simulation.

Condition 2: The action is selected according to the UCB value:

$$a_t^* = \arg\max\nolimits_{a_k \in D_k} [\overline{Q}_t(s, a) + c\sqrt{\tfrac{\ln N(s)}{N(s, a_k)}}]. \tag{14}$$

Since we select the individual action by the UCB value according to Eq.(9) (i.e., Condition 2 is satisfied), we only need to prove Condition 1. At the (t-1)th simulation, let $\vec{a}^{t-1}$ denote the optimal global joint-action returned by Eq.(9), $\overline{u}_{j'}^t$ denote the estimated value of local-action $\vec{a}_{u_{j'}}$ of the utility function $u_{j'}$, and $\overline{Q}_i^t(s, a_i)$ denote the estimated value of agent $i \in Neg_{u_{j'}}$ taking action a_i. According to Eq.(9), the expected value $\overline{Q}_t^i(s, a_i)$ can be updated by

$$\overline{Q}_i^t(s, a_i) = \max\nolimits_{k \in \mathcal{N} \backslash i, a_k \in D_k} \big[\overline{u}_{j'}^t(\vec{a}_{u_{j'}}) + \textstyle\sum_{u_j \in U \backslash u_j'} \overline{u}_j^{t-1}(\vec{a}_{u_j})\big]$$

$$= \max\nolimits_{k \in \mathcal{N} \backslash i, a_k \in D_k} \Big[\overline{u}_{j'}^{t-1}(\vec{a}_{u_{j'}}) + \frac{\sum_{a_l \in u_{j'}} \frac{q_l^{t-1}(a_l)}{Neg_l} - \overline{u}_{j'}^{t-1}(\vec{a}_{u_{j'}})}{N_{u_{j'}}(s, \vec{a}_{u_j'})}$$

$$+ \textstyle\sum_{u_j \in U \backslash u_j} \overline{u}_j^{t-1}(\vec{a}_{u_j})\Big]$$

$$= \max\nolimits_{k \in \mathcal{N} \backslash i, a_k \in D_k} \big[\textstyle\sum_{u_j \in U} \overline{u}_j^{t-1}(\vec{a}_{u_j})$$

$$+ \frac{\sum_{a_l \in u_{j'}} \frac{q_l^{t-1}(a_l)}{Neg_l} + \sum_{u_j, a_l \in Neg_{u_j}} \frac{q_l^{t-1}(a_l)}{Neg_l} - \sum_{u_j \in U} \overline{u}_j^{t-1}(\vec{a}_{u_j}) + K_\epsilon}{N_{u_{j'}}(s, \vec{a}_{u_j'})}\big]$$

$K_\epsilon \in [-(m-1)\epsilon, (m-1)\epsilon]$ is a constant, and m is the number of utility functions. The last equation holds since 1) the Shapley value-based local utility is efficient (i.e., Lemma 2), and 2) the bias of other utility functions $u_j \neq u_{j'}$ is bounded by ϵ. Further, we have $\overline{Q}_i^t(s, a_i)$

$$= \max_{k \in \mathcal{N} \backslash i, a_k \in D_k} \Big[\textstyle\sum_{u_j \in U} \overline{u}_j^{t-1}(\vec{a}_{u_j}) + \frac{V(s, \vec{a}_{t-1}) - \sum_{u_j \in U} \overline{u}_j^{t-1}(\vec{a}_{u_j})}{N_{u_{j'}}(s, \vec{a}_{u'_j})} \Big] \tag{15}$$

$$= \max_{k \in \mathcal{N} \backslash i, a_k \in D_k} \frac{V(s, \vec{a}_{t-1}) + (N_{u_{j'}}(s, \vec{a}_{u'_j}) - 1)(\sum_{u_j \in U} \overline{u}_j^{t-1}(\vec{a}_{u_j}))}{N_{u_{j'}}(s, \vec{a}_{u'_j})}$$

$$= \frac{V(s, \vec{a}_{t-1}) + (N_{u_{j'}}(s, \vec{a}_{u'_j}) - 1)\overline{Q}_i^{t-1}(s, a_i)}{N_{u_{j'}}(s, \vec{a}_{u'_j})} \tag{16}$$

$$= \overline{Q}_i^{t-1}(s, a_i) + \frac{V(s, \vec{a}_{t-1}) - \overline{Q}_i^{t-1}(s, a_i)}{N_{u_{j'}}(s, \vec{a}_{u'_j})}$$

Eq.(15) holds since 1) the constant K_ϵ does not depend on the optimal global-joint action $\max_{k \in \mathcal{N} \backslash i, a_k \in D_k}$, and 2) the cumulative global payoff can be preserved (i.e., Lemma 3), where $V(s, \vec{a}_{t-1})$ is the cumulative global payoffs achieved by the global joint-action $\vec{a}_{t-1}$ at s. Eq.(16) holds since $\max_{k \in \mathcal{N} \backslash i, a_k \in D_k} \sum_{u_j \in U} u_j^{t-1}(\vec{a}_{u_j}) = Q_{t-1}^i(s, a_i)$. Given that the estimated state-action value $\overline{Q}_i^t(s, a_i)$ of agent i satisfies both Conditions 1 and 2, we have that the bias between expected global payoff achieved by the optimal action $\arg\max_{a_i} \overline{Q}_i^T(s, a_i)$ and the optimal global payoff $\overline{Q}^*(s, a^*)$ is $O(\frac{\ln T}{T})$. Since Max-sum can always optimize the global joint action, i.e., each individual agent selects the optimal action with the same global payoff, we can conclude that the bias between the (expected) global payoff of the final global joint-action $\vec{a}$ and the optimal global payoff is bounded by $O(\frac{lnT}{T})$. □

6 EXPERIMENTS

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 use standard errors as the confidence intervals.

6.1 Experimental Domains

We first conduct the experiments in the multi-robot warehouse patrolling in industry chain. Moreover, to improve the generality of our approach, we also conduct additional experiments in traffic signal control and security traffic patrolling for industrial chain.

6.1.1 Multi-Robot Warehouse Patrolling (MRP) [30]. In MRP, multiple mobile robots are deployed to patrol along sensitive regions where persistent surveillance, inspection and control are required. The MRP problem can be formulated as a graph $G = \langle V, E \rangle$, where V is the set of regions 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 for a region at current period is the number of periods elapsed since the last visit. The objective of MRP task is to coordinate patrolling strategies of robots in order to minimize the average idleness of all regions over the whole horizon.

Compared Methods. We compare our coordinated MCTS method with three baselines: 1) **Naive MCTS**, where the global joint-action is explored and evaluated, 2) **FV-MCTS** [9], where the individual action is evaluated according to the global payoff, and 3) **MARL** [30], where each agent adopts a model-free Q-learning to learn the patrolling policy.

6.1.2 Traffic Signal Control (TSC) [43]. We evaluate the proposed methods on the TSC problem on the Cityflow simulation platform [43] within real-world and synthetic traffic networks. 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 two cities [40]. In each traffic network, each vehicle is described as (o, t, d), where o is the origin location (i.e., link), t is time, and d is destination location. The ultimate objective of TSC is to minimize the average travel time of vehicles. In our coordinated MCTS, we model each intersection as an autonomous agent, and use the number of vehicles that exit the traffic network at planning step as the immediate global payoff.

Compared Methods. Besides the online **FV-MCTS**, we compare our coordinated MCTS method with: 1) a traditional transportation method **Maxpressure** [36], which greedily activates the phase with the maximum pressure, 2) **PressLight** [39]: a multiagent reinforcement learning (MARL) method, where each agent learns the policy of choosing next phase by vanilla DQN, and 3) a variant of our coordinated MCTS, where the lookahead depth=1.

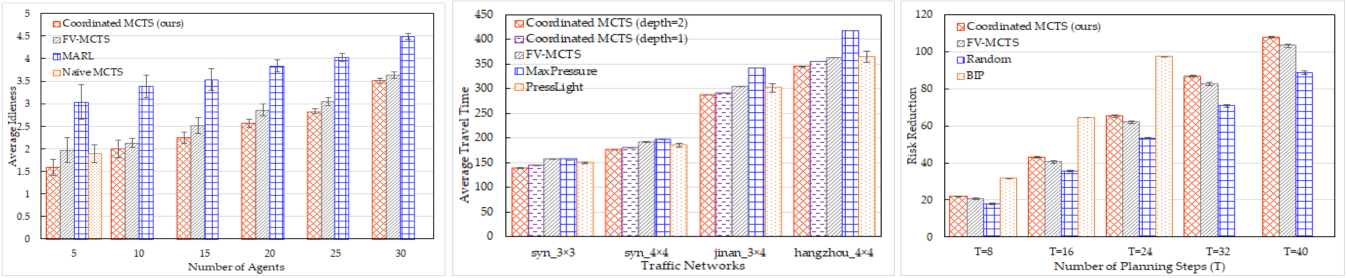
6.1.3 Security Traffic Patrolling (STP) [29]. In STP, a set of police officers (i.e., agents) is deployed for traffic enforcement, with the aim of preventing drivers' illegal behaviors. The interaction between the police and drivers can be cast as a defender-attacker Stackelberg game. The defender (the police) commits to a pure patrol strategy, which is used to generate daily patrol schedules for each police officer. A daily patrol schedule consists of a trajectory through the road network, i.e., a sequence of regions to patrol. The attacker (i.e., drivers) follows the opportunistic behavior model, and reacts to police enforcement in the current and past periods. The risk of accidents measures how likely a serious traffic accident to occur at each region each period. The STP problem is interested in coordinating these police officers' patrolling strategies and minimizing the risks of accidents occurring throughout the game.

Compared Methods. Besides online baselines **Naive MCTS** and **FV-MCTS**, we also compared against a randomized patrolling policy (**Random**) and **Binary integer programming (BIP)** [29], where BIP is used to formulate the pure strategy for traffic enforcement and a master/slave-based optimization is used for scale up.

6.2 Experiment Results

Test the efficiency. Figure 2 shows the efficiency of our coordinated MCTS on improving system performance in different domains. In the MRP domain, we evaluate methods on a synthetic 16 × 16 grid-like graph, and vary the number of agents between 5 and 30. The performance is evaluated on average idleness of nodes. Figure 2(a) shows that our Coordinated MCTS can achieve the minimum average idleness. In the case that there is a smaller number of agents (i.e., 5), Naive MCTS performs better than FV-MCTS. This can be explained by the fact that Naive MCTS can explore promising global joint-actions when team size is small. However, Naive MCTS reaches the timeout even in problems with 10 agents.

In the TSC domain, we evaluate 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



(a) Results on MRP

(b) Results on TSC

(c) Results on STP

Figure 2: Experimental results. LEFT: in MRP, methods are evaluated on average idleness of nodes (the lower the better), MIDDLE: in TSC, methods are evaluated on average travel time of vehicles (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 (i.e., 10 minutes) do not appear.

Traffic Network / Metric		syn_3 × 3		syn_4 × 4		jinan_3 × 4		hangzhou_4 × 4	
		ST (ms)	ATT	ST (ms)	ATT	ST (ms)	ATT	ST (ms)	ATT
#simulation=500	Coordinated MCTS (depth=1)	10.3	144.1	19.8	180.3	13.4	290.9	19.3	355.0
	Coordinated MCTS (depth=2)	10.9	138.4	20.5	176.3	14.4	287.6	20.5	344.7
	Coordinated MCTS (depth=3)	11.1	138.4	20.4	176.1	16.3	287.2	21.0	344.4

Table 1: Test the computation time of our coordinated MCTS in the TSC domain. ST: the computation time (ms) for each simulation step in Algorithm 2, and ATT: the average travel time of vehicles.

on the average travel time of vehicles. From Figure 2(b), we can find that in all networks, compared with online MAP and offline RL baselines, our coordinated MCTS method generates the least average travel time. The potential reason is that for complex road networks with a number of intersections, it is hard for RL methods to train desirable coordination solutions. In contrast, our method enables coordination between neighboring intersections and aims to optimize global traffic. Moreover, within time limits, it is better to use a longer planning horizon (i.e., Coordinated MCTS depth=2) than a shorter planning horizon (i.e., Coordinated MCTS depth=1).

In the STP domain, we evaluate methods on a real-world road network consisting of 284 intersections and 355 roads. The number of agents (i.e., police officers) is set to 30, and the number of planning steps (i.e., T) varies between 8 and 40. The performance is evaluated on the risk (of accidents) reduction between the no-enforcement condition and the provided methods. From Figure 2(c), we can find that compared to FV-MCTS, our coordinated MCTS can reduce the risk by a further 5%. Since BIP returns always the optimal solutions, BIP can reduce the largest risks. However, the complexity of BIP exponentially increases with planning steps (T), preventing it applying to larger problems (e.g., $T \geq 32$).

Test the scalability. In TSC domain (i.e., Figure 2(b)), since the traffic changes in seconds, we set the the time limits for online computation by 5 seconds. In all traffic networks, our coordinated MCTS can return the best solution in an online manner. In MRP and STP domains, since agents need to patrol at a region for some time, we can compute the online plans for next step during patrolling. Thus, in MRP and STP domains, we set the time limits for online computation by 60 seconds. Figure 2(a) and Figure 2(c) show the scalability of our coordinated MCTS with respect to the number of agents n and planning steps (T), from which we can find that our coordinated MCTS and FV-MCTS can apply to large-scale problems with tens of agents (i.e., $n = 30$) and long planning steps (i.e., $T = 40$). In comparison, Figure 2(a) shows that Naive MCTS cannot apply to problems with more than 10 agents, and Figure 2(c) shows that BIP cannot apply to problems with more than 32 planning steps.

Test the computation time. Table 1 shows the computation time of our coordinated MCTS in the TSC domain. From Table 1, 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 (i.e., #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 setting of depth depends both on the time budget and the number of agents. For small budget and large number of agents, the short depth is better, while for large budget and small number of agents, the long depth is better.

7 CONCLUSION

This paper studies the online MAP problem that has a wide range of applications in MASs, and proposes a SCA-based coordinated MCTS method. Our method comprises two parts: Shapley value-based SCA and coordinated MCTS. In the former part, by exploiting structure of DCOP, we can compute Shapley value exactly and efficiently. This Shapley value can be used to provide a fair evaluation of individual action, which is crucial for promoting coordination between agents. In the latter MCTS part, at each state, Shapley value-based SCA is used to decompose the global payoff to each agent. This individual reward backpropagates through the (search) tree to the root state to update local utilities. Experimental results show that combining SCA with MCTS, coordinated MCTS has great advantages on efficiency and scalability.

ACKNOWLEDGMENTS

This research is supported by the National Key Research and Development Program of China under Grant 2022YFB3304400, the National Natural Science Foundation of China under Grant 62076060, 61932007, 71971109, and 62072099, the Key Research and Development Program of Jiangsu Province of China under Grant BE2022157, and the Defense Industrial Technology Development Program under Grant JCKY2021214B002.

REFERENCES

[1] Adrian K. Agogino and Kagan Tumer. 2004. Unifying Temporal and Structural Credit Assignment Problems. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*. 980–987.

[2] Adrian K. Agogino and Kagan Tumer. 2005. Multi-agent Reward Analysis for Learning in Noisy Domains. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. 81–88.

[3] Christopher Amato and Frans A. Oliehoek. 2015. Scalable Planning and Learning for Multiagent POMDPs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. 1995–2002.

[4] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.* 27, 4 (2002), 819–840.

[5] Graeme Best, Oliver M. Cliff, Timothy Patten, Ramgopal R. Mettu, and Robert Fitch. 2019. Dec-MCTS: Decentralized planning for multi-robot active perception. *Int. J. Robotics Res.* 38, 2-3 (2019).

[6] Sushmita Bhattacharya, Siva Kailas, Sahil Badyal, Stephanie Gil, and Dimitri P. Bertsekas. 2020. Multiagent Rollout and Policy Iteration for POMDP with Application to Multi-Robot Repair Problems. In *CoRL'20*. 1814–1828.

[7] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*. 195–210.

[8] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intell. AI Games* 4, 1 (2012), 1–43.

[9] Shushman Choudhury, Jayesh K. Gupta, Peter Morales, and Mykel J. Kochenderfer. 2021. Scalable Anytime Planning for Multi-Agent MDPs. In *AAMAS'21*. 341–349.

[10] Daniel Claes, Frans A. Oliehoek, Hendrik Baier, and Karl Tuyls. 2017. Decentralised Online Planning for Multi-Robot Warehouse Commissioning. In *AAMAS'17*. 492–500.

[11] Daniel Claes, Philipp Robbel, Frans A. Oliehoek, Karl Tuyls, Daniel Hennes, and Wiebe van der Hoek. 2015. Effective Approximations for Multi-Robot Coordination in Spatially Distributed Tasks. In *AAMAS'15*. 881–890.

[12] Aleksander Czechowski and Frans A. Oliehoek. 2020. Decentralized MCTS via Learned Teammate Models. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI'20)*. 81–88.

[13] Yonathan Efroni, Mohammad Ghavamzadeh, and Shie Mannor. 2020. Online Planning with Lookahead Policies. In *NeurIPS'20*.

[14] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R. Jennings. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*. 639–646.

[15] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. 2008. A Linear Approximation Method for the Shapley Value. *Artificial Intelligence* 172, 14 (2008), 1673–1699.

[16] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed Constraint Optimization Problems and Applications: A Survey. *J. Artif. Intell. Res.* 61 (2018), 623–698.

[17] Melvin Gauci, Monica E. Ortiz, Michael Rubenstein, and Radhika Nagpal. 2017. Error Cascades in Collective Behavior: A Case Study of the Gradient Algorithm on 1000 Physical Agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS'17)*. 1404–1412.

[18] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. 2002. Coordinated Reinforcement Learning. In *ICML'02*. 227–234.

[19] Dongge Han, Chris Xiaoxuan Lu, Tomasz Michalak, and Michael Wooldridge. 2022. Multiagent Model-based Credit Assignment for Continuous Control. In *AAMAS'22*. 462–469.

[20] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. In *17th European Conference on Machine Learning (ECML'06)*. 282–293.

[21] Jelle R. Kok and Nikos Vlassis. 2006. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. *J. Mach. Learn. Res.* 7 (2006), 1789–1828.

[22] Adam Lerer, Hengyuan Hu, Jakob N. Foerster, and Noam Brown. 2020. Improving Policies via Search in Cooperative Partially Observable Games. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI'20)*. 7187–7194.

[23] Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. 2021. Shapley Counterfactual Credits for Multi-Agent Reinforcement Learning. In *KDD'21*. 934–942.

[24] Minglong Li, Wenjing Yang, Zhongxuan Cai, Shaowu Yang, and Ji Wang. 2019. Integrating Decision Sharing with Prediction in Decentralized Planning for Multi-Agent Coordination under Uncertainty. In *IJCAI'19*. 450–456.

[25] Qing-Kui Li, Hai Lin, Xi Tan, and Shengli Du. 2020. Consensus for Multiagent-Based Supply Chain Systems Under Switching Topology and Uncertain Demands. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 12 (2020), 4905–4918.

[26] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. 2021. Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning. In *AAMAS'21*. 844–852.

[27] Hang Ma, Jiaoyang Li, T. K. Satish Kumar, and Sven Koenig. 2017. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS'17)*. 837–845.

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*. 4292–4301.

[29] Ariel Rosenfeld, Oleg Maksimov, and Sarit Kraus. 2020. When security games hit traffic: A deployed optimal traffic enforcement system. *Artif. Intell.* 289 (2020), 103381.

[30] Hugo Santana, Geber Ramalho, Vincent Corruble, and Bohdana Ratitch. 2004. Multi-Agent Patrolling with Reinforcement Learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3 (AAMAS'04)*. 1122–1129.

[31] Cleyton R. Silva, Michael Bowling, and Levi H. S. Lelis. 2021. Teaching People by Justifying Tree Search Decisions: An Empirical Study in Curling. *J. Artif. Intell. Res.* 72 (2021), 1083–1102.

[32] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[33] David Silver and Joel Veness. 2010. Monte-Carlo Planning in Large POMDPs. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NeuIPS'10)*. 2164–2172.

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

[35] Zhenggang Tang, Chao Yu, Boyuan Chen, Huazhe Xu, Xiaolong Wang, Fei Fang, Simon Shaolei Du, Yu Wang, and Yi Wu. 2021. Discovering Diverse Multi-Agent Strategic Behavior via Reward Randomization. In *9th International Conference on Learning Representations (ICLR'21)*.

[36] Pravin Varaiya. 2013. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies* 36 (2013), 177–195.

[37] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020. Shapley Q-Value: A Local Reward Approach to Solve Global Reward Games. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI'20)*. 7285–7292.

[38] Wanyuan Wang, Gerong Wu, Weiwei Wu, Yichuan Jiang, and Bo An. 2022. Online Collective Multiagent Planning by Offline Policy Reuse with Applications to City-Scale Mobility-on-Demand Systems. In *21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS'22), Auckland, New Zealand, May 9-13, 2022*. 1364–1372.

[39] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*. 1290–1298.

[40] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. 2019. CoLight: Learning Network-level Cooperation for Traffic Signal Control. In *CIKM'19*. 1913–1922.

[41] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. 2011. Online planning for multi-agent systems with bounded communication. *Artif. Intell.* 175, 2 (2011), 487–511.

[42] Nicholas Zerbel and Logan Yliniemi. 2019. Multiagent Monte Carlo Tree Search. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'19)*. 2309–2311.

[43] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. In *WWW'19*. 3620–3624.