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  
wyyang@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

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

Yuchuan 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. Specifically, 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 joint-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.

* corresponding author

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

ACM Reference Format:  

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 multi-agent 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
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 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 cooperation 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 multiagent 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 $(N, S, A, P, R)$: $N = \{1, 2, \cdots, n\}$ is the set of $n$ agents, $S$ is a
finite set of global states $s$ of the environment. In an MMDP, each agent can observe the global state. $A = A_1 \times \cdots \times A_n$ is the set of joint actions $a = \langle a_1, \ldots, a_n \rangle$. $P(s, a, s') \in \mathcal{P}$ is the transition probability of ending up at state $s'$ given that the joint action $a$ is applied at state $s$. $R(s, a, s') \in \mathcal{R}$ is the immediate global reward for taking the joint action $a$ at state $s$ and ending at state $s'$. In an MMDP, a joint policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$ maps the global state to the global joint-action $a$, and is equivalent to a tuple of individual policies $\pi_i: \mathcal{S} \rightarrow A_i$. Agents interact with environment in sequence for $T$ periods. A finite sequence $\sigma_{\pi} = \langle s_0, s_1, \ldots, s_T \rangle$ of states generated by a policy $\pi$ is called a trajectory. For any joint policy $\pi$ on an MMDP, the expected discounted cumulative global reward in state $s$ is $V^\pi(s) = \mathbb{E}_{\pi} [\sum_{k=0}^{T-1} \gamma^k R(s_k, \alpha_k, s_{k+1}) | s_0 = s]$, where $\mathbb{E}$ denotes the expected value by following $\pi$, and $\gamma \in [0, 1]$ is the discount factor. The objective for solving an MMDP is to find a joint policy $\pi$ that can generate a trajectory $\sigma_{\pi}$ 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 SHAPELEY 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 $G = (N, A, D, U)$ such that: $N = \{1, 2, \ldots, n\}$ is the set of agents, $A = \{a_1, a_2, \ldots, a_n\}$ is the set of variables, each $a_i$ is controlled by an agent $i$, and $t$ takes the value from the finite discrete domain $D_i \in D$, and $U = \{u_1, u_2, \ldots, u_m\}$ is a set of local utilities, each $u_j$ is defined as a mapping from the assignments of the involved k variables $\bar{a}_{u_j} = (a_{j_1}, \ldots, a_{j_k})$ to a positive real, i.e., $u_j: D_{j_1} \times D_{j_2} \times \cdots \times D_{j_k} \rightarrow \mathbb{R}_{\geq 0}$. Let $N_u$ denote the set of agents involved in the utility $u_j$. Without loss of generality, let $noop$ denote the no operation action, and $\forall u_j, u_j(noop) = 0$, i.e., there will be no utilities if all agents do nothing. A solution to a DCOP is to find a global joint action $\bar{a}$ that maximizes the global payoff $u(N) = \sum_{u_j \in U} u_j(\bar{a}_{u_j})$, which is the sum of all local utility functions, i.e., $\bar{a} = \arg \max_{a} u(N)$.

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 $G = (N, A, D, U)$, 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 $i$, $N_{loc}^i = \{k | \exists u_j \in U : i \in N_u \wedge k \in N_u\}$.

**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 $N_{loc}^1 = \{2\}$. Similarity, we have $N_{loc}^2 = \{1, 3, 4\}$, $N_{loc}^3 = \{2\}$, and $N_{loc}^4 = \{2\}$.

**Max-sum for Optimizing DCOP.** Given a DCOP, Max-sum algorithm has been proposed for optimizing the global joint action $\bar{a}$ 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 $\{\}$ (resp. Neg $\{\}$) 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 $a_i \in D_i$:

$$E_{i \rightarrow u_j}(a_i) = \sum_{u_j \in Neg_{u_j} \wedge \exists \alpha_{a_j} \in u_j} F_{u_j \rightarrow \alpha_i}(a_i) - \alpha.$$ (1)
Algorithm 1: Max-sum (0)

```
Input : The DCOP $\langle N, A, D, U \rangle$.
Output: Joint action $\pi^*$. 
1 for $k$ iterations do 
2 for variable node $i$ do 
3 produce message $E_i \rightarrow u_j$ using messages from $N e_j \setminus \{u_i\}$; 
4 for function node $u_j$ do 
5 produce message $F_{u_j} \rightarrow i$ using messages from $N e_i \setminus \{i\}$; 
6 Compute the joint action $\pi^*$ by Eq.(3);
```

where $F_{u_j} \rightarrow i(a_j)$ is the utility for value $a_j$ included in the messages received from the function node $u_j$ at iteration $k = 1, \alpha$ 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 $a_j \in D_j$ the maximal utility of any combination of assignments to the variables involved in $u_j$ apart from $i$. Formally, the message from function node $u_j$ to variable $i$ includes for each value $a_i \in D_i$:

$$F_{u_j} \rightarrow i(a_j) = \max_{a_i} \left( \arg \max_{a_j} \Delta_{\pi} \left[ u_j(a_i, a_j) + \sum_{i \in N e_j, i \neq j} E_i \rightarrow u_j(a_i) \right] \right).$$

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

$$a_i^* = \arg \max_{a_i} \sum_{j \in N e_i} F_{u_j} \rightarrow i(a_j).$$

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 $\pi^*$ 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 $\mathcal{G} = \langle N, A, D, U \rangle$ and a global joint-action $\alpha = \{a_1, \cdots, a_n\}$ returned by Max-sum, let $u(C) = \sum_{j: a_j \neq a_j^*} u_j(C)$ denote the total utility achieved by the coalition $C \subseteq N$, where $a_j^* = a_j$ if the agent $i \in C$, $a_j^* = \text{noop}$ otherwise. The marginal contribution $\Delta^\pi_i$ of the agent $i$ to a coalition $C \subseteq N \setminus \{i\}$ is the increase in the utility of $C$ as a result of $i$ joining it and taking action $a_i$, i.e.,

$$\Delta^\pi_i(C) = u(C \cup \{i\}) - u(C).$$

Shapley Value. The Shapley value of each agent $i$ taking action $a_i$, $\phi^\pi_i$ is the average of its marginal contribution to all coalitions:

$$\phi^\pi_i = \frac{1}{n!} \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(n-|C|-1)!}{n!} \Delta^\pi_i(C).$$

The Shapley value can be interpreted as that all agents are arranged in some order, all orderings being equally likely, and then $\phi^\pi_i$ 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 $\phi^\pi_i$. 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 $\mathcal{G} = \langle N, A, D, U \rangle$, let $\bar{\alpha}$ denote any global joint-action, then the global payoff generated by the grand coalition $N$ is equal to the sum of the Shapley value-based credits of all agents, i.e., $\sum_i \phi^\pi_i = u(N)$.

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’s action has an effect only on the utilities of the coalition of the variable 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 $\bar{\alpha} = \{1, 2, 1\}$, where $u_1(a_1, a_2) = 20, u_2(a_2, a_3) = 15, u_3(a_2, a_3) = 12$. 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., $\Delta^\pi_1(C_i) = 0$ (1 $\leq i \leq 3$).

On the other hand, for these relevant coalitions $C_1 = \{2\}$, $C_5 = \{2, 3\}$, 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., $\Delta^\pi_1(C_i) = u(C_i \cup \{i\}) - u(C_i) = 18$ (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 $N^0_i$ denote the subset of the locally interacted agents (defined in Definition 1) and $n^0_i = |N^0_i|$, and $N^\text{irr}_i$ denote the subset of irrelevant agents that do not interact with agent $i$ and $n^\text{irr}_i = |N^\text{irr}_i|$. The Shapley value of $i$, $\phi^\pi_i$ then can be rewritten by

$$\phi^\pi_i = \frac{1}{n!} \sum_{C \subseteq N^0_i} H(|C|, n^\text{irr}_i) \Delta^\pi_i(C).$$

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

$$H(|C|, n^\text{irr}_i) = \sum_{k=0}^{n^\text{irr}_i} \binom{n^\text{irr}_i}{k} ((|C| + k)!/(n - |C| - k - 1)!).$$

**Proof.** For each subset of agents $C$, dividing it into disjoint coalitions of locally interacted agents $C_1 \subseteq N^0_i$ and irrelevant
agents $C_i \subseteq N^{irr}_i$ of agent $i$, we have
$$q_i^{\hat{d}} = \frac{1}{n_i} \sum_{C \subseteq N^{loc}_i, \gamma \neq \epsilon} C \mid |n_i - |C_i| - |C| - 1| \cdot g_i^{\hat{d}}(C).$$

$$= \frac{1}{n_i} \sum_{C \subseteq N^{loc}_i} \sum_{C \subseteq N^{irr}_i} |C| \cdot C \mid |n_i - |C_i| - |C| - 1| \cdot g_i^{\hat{d}}(C_i)$$

$$= \frac{1}{n_i} \sum_{C \subseteq N^{loc}_i} \sum_{C \subseteq N^{irr}_i} |C| \cdot C \mid |n_i - |C_i| - |C| - 1| \cdot g_i^{\hat{d}}(C_i)$$

$$= \frac{1}{n_i} \sum_{C \subseteq N^{loc}_i} \sum_{C \subseteq N^{irr}_i} (n_i^{irr} - k) \cdot (n_i - |C_i| - k - 1) \cdot g_i^{\hat{d}}(C_i).$$

The reason that Eq. (8) holds is that given a local coalition $C_i \subseteq N^{loc}_i$, for any irrelevant coalition $C_j \subseteq N^{irr}_j$, the agent $i$ has the same marginal contribution to such a joint coalition $C_i \cup C_j$, that is $\forall C_i, C_j \subseteq N^{irr}_i$, $\Delta g_i^{\hat{d}}(C_i \cup C_j) = \Delta g_i^{\hat{d}}(C_i \cup C_j)$. □

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 $H(x, y)\ for 1 \leq y \leq n, 0 \leq x \leq n - y$ 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 $G(s) = (N, A, D, U)$. On the one hand, agents need to coordinate their joint-action $\tilde{a}$ 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 procedure (i.e., Eq. (3)).

$$q_i(a_i) = \arg\max_{a_i} \left[ \frac{\sum_{j \in Neg_i} F_{a_i \rightarrow s}(a_i) + c \sqrt{\ln N(s) / N_{i}(s, a_i)}}{N_{i}(s, a_i)} \right].$$

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.

**Updating Individual Action and Local Utility Function.** Given the DCOP $G(s) = (N, A, D, U)$ at state $s$, let $\tilde{a} = (a_1, \ldots, a_n)$ denote the global joint-action, where $a_i$ is selected by Eq. (9), and $R(s, \tilde{a}, s')$ denote the immediate global reward achieved by taking $\tilde{a}$ and ending the state $s'$. We first use the Shalev value to determine the immediate individual reward, i.e., $\forall i, R_i(s, a_i, s') = \phi_i^{\hat{d}}$. In coordinated MCTS, the key is to model each local utility function $u_j \in U$ 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.,

$$q_i(s, a_i) = \sum_{j \in Neg_{u_j}(s)} q_i(s, a_i).$$

**Algorithm 2: SCA-guided Coordinated MCTS**

**Input:** time limit, depth, $c$, state $s$, discounted factor $\gamma$.

**Output:** Joint Action $\tilde{a}$. 

**while** time limit not reached **do**

1. **Simulate**($s$, depth);

2. $\tilde{a} \leftarrow$ Max-sum(0) by Algorithm 1;

3. **Function Simulate**($s$, depth):

4. **if** depth = 0 **then**

5. **return** 0;

6. **InitializeState**($s$);

7. $\tilde{a} \leftarrow$ Max-sum($c$), $s' \sim P(s, \tilde{a}, s')$;

8. **for** each agent $i \in N$ **do**

9. $R_i(s, a_i, s') = \phi_i^{\hat{d}}$ via Eq. (6);

10. $q_i(s, a_i)$ **←** $R_i(s, a_i, s') + \gamma \cdot \text{Simulate}(s', \text{depth} - 1), R_i$;

11. **UpdateState**($s, \tilde{a}, \tilde{u}$);

12. **Function InitializeState**($s$):

13. **if** $s$ is a new state **then**

14. Formulate $G(s) = (N, A, D, U)$;

15. **for** each agent $i \in N$ **&** $a_i \in D_i$ **do**

16. Initialize $N(s, a_i) = 0$;

17. **for** each utility function $u_j \in U$ **&** $\tilde{a}_{u_j} \in D_{u_j}$ **do**

18. Initialize $\tilde{u}_j(s, \tilde{a}_{u_j}) = 0$, and $N_{u_j}(s, \tilde{a}_{u_j}) = 0$;

19. **Function UMax-Sum($c$):**

20. Steps 1-5 of Max-sum($0$) by Algorithm 1;

21. Compute the global joint-action $\tilde{a}$ by Eq. (9);

22. **Function UpdateState**($s, \tilde{a}, \tilde{u}$):

23. **for** each agent $i \in N$ **do**

24. $N_i(s, a_i) + 1$;

25. **for** each utility function $u_j \in U$ **do**

26. $N_{u_j}(s, \tilde{a}_{u_j}) + 1$, $\tilde{u}_j(s, \tilde{a}_{u_j}) = \sum_{i \in Neg_{u_j}(s)} q_i(s, a_i)$ \frac{q_i(s, a_i)}{N_{i}(s, a_i)}$;

27. $\tilde{u}_j(s, \tilde{a}_{u_j}) + \frac{q_i(s, a_i)}{N_{i}(s, a_i)}$ \frac{q_i(s, a_i)}{N_{i}(s, a_i)}$;

28. $u_j(s, \tilde{a}_{u_j})$ **←** \frac{q_i(s, a_i)}{N_{i}(s, a_i)}$ \frac{q_i(s, a_i)}{N_{i}(s, a_i)}$. 

547
where $Neg_u(s)$ denotes the set of neighbor agents of $u_i$ and $Neg_i(s)$ denotes the neighbors of the agent $i$ at current DCOP $G(s)$. This way of updating local utility function can preserve the efficiency property of Shapley value, shown as follows.

**Lemma 2.** Given a DCOP $G = (N, A, D, U)$, let $\bar{a}$ denote any global joint-action, $\phi^a_i$ denote the Shapley value of agent $i$, and $q_{ui}(\bar{a}_{ui}) = \sum_{i \in Neg_u} \frac{\phi^a_i}{|Neg_u|}$ 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., $u(N) = \sum_{u_i \in U} q_{ui}(\bar{a}_{ui})$.

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

$$\bar{\pi}(s, \bar{a}_{ui}) = \bar{\pi}(s, \bar{a}_{ui}) + \frac{q_{ui}(s, \bar{a}_{ui}) - \bar{\pi}(s, \bar{a}_{ui})}{N_{ui}(s, \bar{a}_{ui})},$$

where $N_{ui}(s, \bar{a}_{ui})$ is the visit frequency of the local joint-action $\bar{a}_{ui}$. The local utility statistics $\bar{\pi}(s, \bar{a}_{ui})$ represent the cumulative long-term utility function, which will be used in messsage 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** (Steps 13-19): for a new state $s$, we first build the DCOP model $G(s) = (N, A, D, U)$ (Step 15). We then initialize the frequency statistics of each individual action $N_i(s, a_i)$ and local joint-action $N_{ui}(s, \bar{a}_{ui})$, and the mean value of the utility $\bar{\pi}(s, \bar{a}_{ui})$ for each local joint action $\bar{a}_{ui}$ (Steps 16-19).

**Simulate** (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** is used to select the global joint-action $\bar{a}$, and transit to state $s'$. Given the global joint-action $\bar{a}$, we use the Shapley value to determine the individual reward for each agent $i$, i.e., $R_i(s, a_i, s') = \phi^a_i$ (Step 10). At Step 11, by further simulation, the cumulative state-action reward $q(s, a_i)$ of agent $i$ is computed by summing the discounted future individual reward $\gamma \cdot \text{Simulate}(s', \text{depth} - 1), R_i$, where $R_i$ indicates the individual reward of agent $i$ returned at state $s'$.

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

### 5.2 Theoretical Analysis

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

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

$$V^\rho(s_0, \bar{a}_0) = \sum_{u_i \in U} q_{ui}(s_0, \bar{a}_{ui}, s_1)$$

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 $u^*(N)$ denote the optimal expected global payoff and $\mathbb{E}[\bar{u}(N)]$ 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 $\bar{u}_j$ and its real value is bounded by $\epsilon$, i.e., $\forall u_j \neq u_f, \bar{u}_j \in D_{ui}, |\mathbb{E}[\bar{\pi}_j(\bar{a})] - u^*_j(\bar{a})| \leq \epsilon$, the bias between $u^*(N)$ and $\mathbb{E}[\bar{u}(N)]$ is $\mathbb{E}[\bar{u}(N)] - u^*(N) = O(\ln T/T)$, where $T$ is the number of simulations at state $s$.

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

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

$$Q^*_t(s, a) = Q^*_{t-1}(s, a) + \frac{\nu(s, a) - Q^*_{t-1}(s, a)}{N(s, a)}$$

where $\nu(s, a)$ 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:

$$a^*_t = \arg \max_{a \in D} [Q^*_t(s, a) + c \sqrt{\frac{\ln T}{N(s, a)}}]$$

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 $Q^{t-1}_{\gamma}$ denote the optimal global joint-action returned by Eq.(9), $\bar{\pi}^{t-1}_{\gamma}$ denote the estimated value of local-action $\bar{a}_{ui}$ of the utility function $u_j$, and $\bar{Q}^{t-1}_{\gamma}(s, a_i)$ denote the estimated value of agent $i \in Neg_{ui}$ taking action $a_i$. According to Eq.(9), the expected value $Q^*_t(s, a_i)$ can be updated by

$$Q^*_t(s, a_i) = \max_{a \in D} [Q^*_{t-1}(\bar{a}_{ui}) + \sum_{u_j \in U \backslash u_f} \bar{\pi}^{t-1}_{\gamma}(\bar{a}_{ui})]$$

$$= \max_{a \in D} [Q^*_{t-1}(\bar{a}_{ui}) + \sum_{u_j \in U \backslash u_f} \bar{\pi}^{t-1}_{\gamma}(\bar{a}_{ui})]$$

$$= \max_{a \in D} [Q^*_{t-1}(\bar{a}_{ui}) + \sum_{u_j \in U \backslash u_f} \bar{\pi}^{t-1}_{\gamma}(\bar{a}_{ui})]$$

$$= \max_{a \in D} [Q^*_{t-1}(\bar{a}_{ui}) + \sum_{u_j \in U \backslash u_f} \bar{\pi}^{t-1}_{\gamma}(\bar{a}_{ui})]$$

$$+ \sum_{a \in D} \mathbb{E}[\bar{u}(s, a)] - \sum_{a \in D} \mathbb{E}[\bar{u}(s, a)] + K_c$$

where $K_c \in [-\infty, 0]$ 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 \( \epsilon \). Further, we have
\[
\mathcal{Q}_{1}^f(s, a_1) = \max_{k \in \mathcal{N} \setminus i, a_k \in \mathcal{D}_k} \left[ \sum_{u_j \in \mathcal{U}} \frac{u_j^{-1}(\tilde{a}_{uj})}{N_{u_j}(s, \tilde{a}_{uj})} + \frac{V(s, \tilde{a}_{i-1}) - \sum_{u_j \in \mathcal{U}} u_j^{-1}(\tilde{a}_{uj})}{N_{u_j}(s, \tilde{a}_{uj})} \right]
\]
\[
= \max_{k \in \mathcal{N} \setminus i, a_k \in \mathcal{D}_k} \frac{V(s, \tilde{a}_{i-1}) + (N_{u_j}(s, \tilde{a}_{uj}) - 1)\mathcal{Q}_{1}^f(s, a_i)}{N_{u_j}(s, \tilde{a}_{uj})} = \mathcal{Q}_{1}^f(s, a_1)
\]
\[
= \mathcal{Q}_{1}^f(s, a_1) = \max_{a_1 \in \mathcal{A}_i} \mathcal{Q}_{1}^f(s, a_1) = \mathcal{Q}(s, a_1)
\]

Eq.(15) holds since 1) the constant \( K_c \) does not depend on the optimal global-joint-action max \( \max_{i \in \mathcal{N}, a_i \in \mathcal{D}_i} \) and 2) the cumulative global payoff can be preserved (i.e., Lemma 3), where \( V(s, \tilde{a}_{i-1}) \) is the cumulative global payoffs achieved by the global joint-action \( \tilde{a}_{i-1} \) at \( s \). Eq.(16) holds since max \( \max_{i \in \mathcal{N}, a_i \in \mathcal{D}_i} \sum_{u_j \in \mathcal{U}} u_j^{-1}(\tilde{a}_{uj}) = \mathcal{Q}_{1}^f(s, a_1) \). Given that the estimated state-action value \( \mathcal{Q}_{1}^f(s, a_1) \) of agent \( i \) satisfies both Conditions 1 and 2, we have that the bias between expected global payoff achieved by the optimal action arg \( \max_{u_i} \mathcal{Q}_{1}^f(s, a_i) \) and the optimal global payoff \( \mathcal{Q}(s, a') \) is \( O(\frac{1}{e_i}) \). 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 \( \tilde{a} \) and the optimal global payoff is bounded by \( O(\frac{1}{e_i}) \).

\[\square\]

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 \( G = (V, E) \), where \( V \) is the set of regions and \( e_{ij} \in E \) denotes the regions \( u_i \) and \( u_j \) 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 idle time 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 \( \text{syn}_3 \times 3 \) and \( \text{syn}_4 \times 4 \) are generated by the Cityflow. Two representative real-world traffic datasets \( \text{jianan}_3 \times 4 \) and \( \text{hangzhou}_4 \times 4 \) 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 multi-agent 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 improved 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 \( 16 \times 16 \) grid-like graph, and vary the number of agents between 5 and 30. The performance is evaluated on average idle time of nodes. Figure 2(a) shows that our Coordinated MCTS can achieve the minimum average idle time. 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 \( \text{syn}_3 \times 3 \) and \( \text{syn}_4 \times 4 \), and two real-world road networks \( \text{jianan}_3 \times 4 \) and \( \text{hangzhou}_4 \times 4 \). The performance is evaluated
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.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Traffic Network</th>
<th>syn_3 × 3</th>
<th>syn_4 × 4</th>
<th>jinan_3 × 4</th>
<th>hangzhou_4 × 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coordinated MCTS (depth=1)</td>
<td>ST (ms)</td>
<td>ATT</td>
<td>ST (ms)</td>
<td>ATT</td>
</tr>
<tr>
<td>#simulation=500</td>
<td>Coordinated MCTS (depth=2)</td>
<td>10.3</td>
<td>144.1</td>
<td>19.8</td>
<td>180.3</td>
</tr>
<tr>
<td></td>
<td>Coordinated MCTS (depth=3)</td>
<td>10.9</td>
<td>138.4</td>
<td>20.5</td>
<td>176.3</td>
</tr>
</tbody>
</table>

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.

In conclusion, the setting of depth depends both on the time budget (i.e., #simulation=500) and the number 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 scalability. In TSC domain (i.e., Figure 2(b)), since the traffic changes in seconds, we set 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).

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.
REFERENCES


