Nucleolus Credit Assignment for Effective Coalitions in Multi-agent Reinforcement Learning

Yugu Li
University of South Australia
Adelaide, Australia
liyyy301@mymail.unisa.edu.au

Zehong Cao*
University of South Australia
Adelaide, Australia
jimmy.cao@unisa.edu.au

Jianglin Qiao
University of South Australia
Adelaide, Australia
jianglin.qiao@unisa.edu.au

Siyi Hu
University of South Australia
Adelaide, Australia
siyi.hu@unisa.edu.au

ABSTRACT

In cooperative multi-agent reinforcement learning (MARL), agents typically form a single grand coalition based on credit assignment to tackle a composite task, often resulting in suboptimal performance. This paper proposed a nucleolus-based credit assignment grounded in cooperative game theory, enabling the autonomous partitioning of agents into multiple small coalitions that can effectively identify and complete subtasks within a larger composite task. Specifically, our designed nucleolus Q-learning could assign fair credits to each agent, and the nucleolus Q-operator provides theoretical guarantees with interpretability for both learning convergence and the stability of the formed small coalitions. Through experiments on Predator-Prey and StarCraft scenarios across varying difficulty levels, our approach demonstrated the emergence of multiple effective coalitions during MARL training, leading to faster learning and superior performance in terms of win rate and cumulative rewards especially in hard and super-hard environments, compared to four baseline methods. Our nucleolus-based credit assignment showed the promise for complex composite tasks requiring effective sub-teams of agents.

KEYWORDS

MARL; Credit assignment; Nucleolus Q learning; Multiple small coalitions

ACM Reference Format:
Yugu Li, Zehong Cao, Jianglin Qiao, and Siyi Hu. 2025. Nucleolus Credit Assignment for Effective Coalitions in Multi-agent Reinforcement Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

In multi-agent environments, determining the contribution of each agent from reward signals is critical for effective cooperation to complete a composite task [1, 7, 15]. In cooperative multi-agent reinforcement learning (MARL), this process, known as Credit Assignment [10], is central to improving both the performance and interpretability of MARL systems. Credit assignment approaches for MARL can be broadly classified into two categories: implicit and explicit methods.

Implicit credit assignment methods, such as VDN [31], QMIX [23], and WQMIX [22], rely on value decomposition to indirectly infer the contribution of each agent from the learning process. While these methods scale efficiently, their reliance on predefined value decomposition structures often leads to weak performance, when the task environment does not align well with these assumptions. Moreover, they have limited interpretability, making it difficult to understand the specific contributions of individual agents during task execution [6].

Explicit credit assignment methods, on the other hand, directly evaluate each agent's contribution based on its actions, providing better interpretability and robust performance. Benchmark algorithms in this category include COMA [8], SQDDPG [34], and SHAQ [33]. COMA, an early approach, employs a centralized critic with a counterfactual baseline to assess each agent's action value. However, its ability to accurately attribute contributions from individual agents becomes restricted in complex environments. More recent methods, such as SQDDPG and SHAQ, leverage Shapley value, a concept from cooperative game theory, to fairly distribute rewards based on each agent's contribution. These methods offer strong theoretical convergence guarantees and improved interpretability by evaluating the individual contributions of agents [11].

However, both implicit and explicit credit assignment methods predominantly assume that all agents form a single grand coalition to tackle a composite task. While promoting fairness and stability, they occasionally lead to inefficiencies, especially in real-world applications that involve multiple, smaller sub-teams of agents working on different subtasks. For instance, tasks such as resource management and emergency response (e.g., fire rescue) require agents to form smaller, dynamic coalitions that can cooperate on subtasks while still contributing to the overall mission [2, 5, 13]. But existing credit assignment fails to incentivize the formation of these smaller coalitions, resulting in larger search spaces and reduced learning efficiency [20]. To address this challenge, it is essential to enable the emergence of smaller, effective coalitions for cooperative

**Corresponding author

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

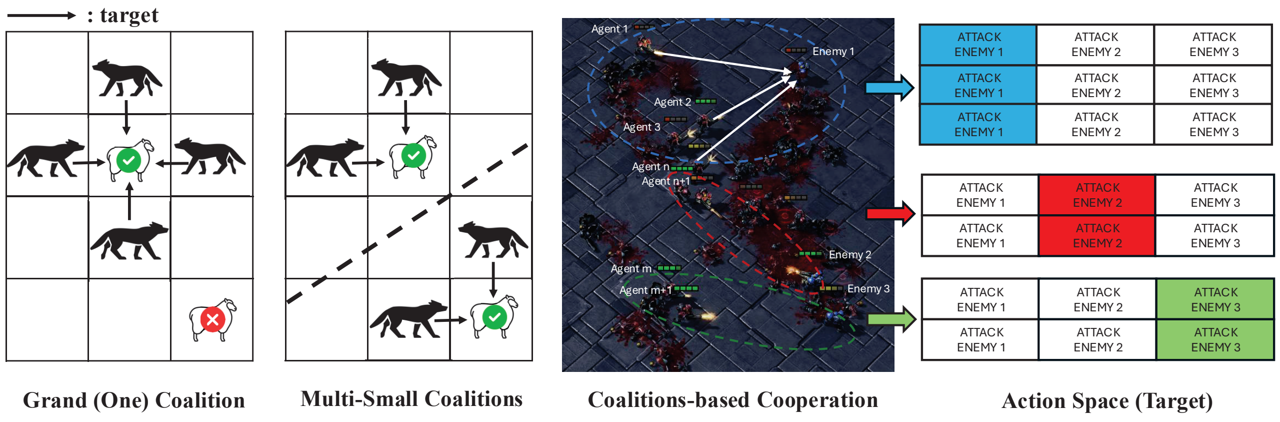


Figure 1: The transition from a single grand coalition to multiple smaller, task-specific coalitions in MARL. In scenarios like SMAC in super-hard maps: where a large number of agents are involved, forming multiple small coalitions is crucial for task completion efficiently. Agents who attack the same enemy unit naturally form these coalitions, enabling them to work together efficiently to achieve the mission.

MARL. We expect these small coalitions can only cooperate on subtasks, improving overall composite task performance and learning efficiency, as shown in **Fig. 1**. Agents can transit from a single grand coalition to multiple smaller, task-specific coalitions in MARL, and ensure that these smaller coalitions remain fair, interpretable, and stable over time.

In this paper, we propose a novel credit assignment approach based on the nucleolus concept from cooperative game theory [27]. The nucleolus is designed to minimize the maximum dissatisfaction among agents by providing fair reward allocations across multiple small coalitions rather than a single grand coalition. *Fairness* is achieved by calculating each agent's contribution relative to what each coalition could achieve independently, ensuring that no agent feels undervalued. In doing so, the nucleolus encourages agents to remain committed to their coalitions. Without fair reward allocation, agents may become dissatisfied and leave their coalitions. Nucleolus also improves *interpretability* by explicitly linking reward allocation to the agents' contributions, making it easier to trace why a particular coalition receives a given reward. *Stability* is guaranteed as the nucleolus works to minimize excess—the gap between a coalition's potential independent gain and its current reward—thereby ensuring that no coalition has the incentive to deviate or reorganize, maintaining consistent cooperation throughout the task. Thus, our introduced nucleolus-based credit assignment for MARL will form stable and fair small coalitions. These coalitions partition a composite task into manageable subtasks, significantly improving both learning efficiency and MARL system performance. Additionally, we provide theoretical guarantees for the convergence of the proposed nucleolus-Q operator, ensuring that the action values of agents within each coalition converge to a local optimum. Through experiments on standard benchmarks such as Predator-Prey and StarCraft Multi-Agent Challenge, we demonstrate that our approach outperforms the existing four methods in terms of learning efficiency and overall performance.

2 RELATED WORK

2.1 Credit Assignment in CTDE

Implicit Credit Assignment. Credit assignment is a fundamental challenge in Centralized Training with Decentralized Execution (CTDE) [18]. VDN [31] introduces an additivity assumption, where the global Q-value is assumed to be the sum of the local Q-values of individual agents. This decomposition allows for simpler representation but significantly limits the expressiveness of the global Q-value. To overcome this issue, QMIX [23] extends VDN by developing a mixing network that allows non-linear combinations of local Q-values to form the global Q-value. This non-linearity enhances the representative capacity of the global Q-value but remains limited by its implicit credit assignment mechanism, where agent contributions are not explicitly modeled. Consequently, QMIX struggles with environments featuring non-monotonic rewards. WQMIX [22] further looks at this issue by introducing a learnable weighting mechanism to dynamically adjust the contributions of each agent's local Q-value in the global Q-value computation. While WQMIX improves adaptability, the selection of weighting coefficients often relies on trial and error, with minimal interpretability. Moreover, improper weight adjustments can lead to instability in the learning process, further complicating its application in complex multi-agent environments.

Explicit Credit Assignment. For explicit credit assignment methods, they can directly compute the contribution of each individual agent, ensuring that each agent's policy is optimized locally. One such method is COMA [8], which discovers a counterfactual baseline to evaluate the difference in global reward when an agent takes alternative actions. COMA calculates each agent's contribution to the current action by observing how global Q-value changes, but it suffers from inefficiency issues. Because the counterfactual baseline is derived from the current joint action rather than the optimal one, this credit assignment may not update to an optimal joint policy. Additionally, the counterfactual calculation in COMA increases computational complexity and reduces sample efficiency. More recently, methods such as Rollout-based Shapley Values [25], SQDDPG [34], and SHAQ [33] have leveraged the Shapley value [28] from cooperative game theory to improve the interpretability of credit assignment in MARL. Shapley value-based methods offer a principled way to fairly distribute rewards based on each agent's contribution. While SQDDPG builds on the DDPG network architecture and adopts a centralized approach similar to COMA, it is difficult for multi-agent Q-learning. SHAQ employs a decentralized approach, allowing for more scalable credit assignment, but these Shapley value-based methods assume a single grand coalition. In scenarios that require multi-agent cooperation based on multiple, smaller coalitions, the aforementioned explicit credit assignment methods do not achieve efficiency.

2.2 Nucleolus in Cooperative Games

In game theory, the Nucleolus concept [27] is a solution to identify the most equitable allocation of resources or payoffs in a cooperative game by minimizing the maximum dissatisfaction (or excess) among coalitions. For example, Gautam et al. [9] highlight the role of the nucleolus in power and energy systems, emphasizing its effectiveness in resolving resource allocation, demand response, and system reliability issues. Similarly, Oner & Kuyzu [35] proposes nucleolus-based cost allocation methods and develops column and row generation techniques to solve the constrained lane covering game for optimizing truckload transportation procurement networks, which uses a nucleolus approach to minimize costs and fairly allocate them among shippers. These approaches are generally tailored for static reward settings with well-understood environments, rather than for maximizing rewards in MARL contexts. In dynamic environments, where rewards are delayed, traditional nucleolus-based methods fall short, necessitating the development of more flexible exploration policies for agents to adapt to uncertain conditions.

3 METHOD

We begin by developing an entity-based, partially observable coalition Markov decision process that supports coalition formation in Subsection 3.1. Next, our proposed nucleolus-based credit assignment in Subsection 3.2: (i) define the nucleolus-Q value by incorporating the concept of the nucleolus into Q-learning to ensure both the optimal coalition structure and optimal actions, supported by a theorem proof, and (ii) introduce a new nucleolus-based Bellman operator that converges to the optimal nucleolus Q-value, also with a corresponding theorem proof.

3.1 Entity-based Coalition POMDP

We extend Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework [14, 16, 17] by defining an Entity-based Coalition POMDP, named as EC-POMDP. Here, entities include both controllable agents and other environment landmarks, and it is essential to distinguish the terms between a composite task and its subtasks: a subtask represents the smallest unit of reward feedback in the environment and cannot be further divided, whereas the collection of all subtasks is referred to as a composite task. Formally, the EC-POMDP framework is represented as $\langle S, N, A, P, O, \gamma, E, G, (E_g, A_g, r_g)_{g\in G}, R, CS\rangle$, where S denotes a set of states of the environment; $N = \{1, \dots, n\}$ represents a set of agents; $A = \times_{i\in N} A_i$ is a set of joint actions of agents, each joint action $a = (a_1, \dots, a_n) \in A$, where a_i represents the action of agent $i \in N$; P is the state transition function defined as $P(s'|s, a)$; $o_i \in O$ is the agent's observation; γ represents the learning rate; E is set of entities, where each entity $e \in E$ has state $s^e \in S$ and let $S^e \subseteq S$ define all possible states of entity e; $G = \{g_1, \dots, g_m\}$ is the composited task of the environment with m subtasks; each subtask g is defined as tuple $(E_g, A_g, r_g)_{g\in G}$, where $E_g \subseteq E$ is a set of subtask-specific entities with their states $S^{E_g} = \bigcup_{e\in E_g} S^e$, $A_g \subseteq A$ representing actions used to complete subtasks g and $r_g : S^{E_g} \times S \times A_g \to \mathbb{R}$ is the reward function to complete sub-task g; R is the total reward for the composite task G; $CS = \{C_g \subseteq N | \forall g \in G\}$ is the coalition structure, which satisfied with $\bigcup_{g\in G} C_g = N$ and $C_g \cap C_{g'} = \emptyset$ for all $g, g' \in G$, and each $C_g = \{i | \forall i \in N, s.t. a_i \in A_g\}$ is a coalition of agents that actions of agents is needed for each subtask g.

EC-POMDP is a Markov decision process that evolves continuously over discrete time steps. We assume that, in the initial state, all agents are in a single grand coalition status. As subtasks are identified during the process, this grand coalition breaks into multiple smaller coalitions, each dedicated to completing specific subtasks. Upon completing these subtasks, these smaller coalitions will reconsolidate into a grand coalition to complete the composite task finally.

3.2 Nucleolus-based Credit Assignment

Nucleolus in Cooperative Game. The nucleolus is a concept of fairness used to solve payoff distribution in cooperative games. It focuses on allocating payoffs among coalitions to minimize the maximum dissatisfaction (excess), thereby achieving a stable and fair distribution [27]. It provides stability and predictability, as it is a unique and definitive solution in every game [21]. The nucleolus is typically located within or near the core of the game, ensuring the stability of the distribution scheme, and it balances the interests of different coalitions to reduce the likelihood of unfair allocation. This approach is widely applied in fields such as economics, politics, and supply chain management to benefit distribution in cooperation settings.

Let (N, u, v) be a cooperative game [4], where $N = \{1, \dots, n\}$ is the set of agents, $u(C)$ is the unity function to measure the profits earned by a coalition $C \subseteq N$, v represents the total assets that can be distributed. Let $x = (x_1, \dots, x_n)$ be a payoff distribution over N. For a coalition C, the excess $e(C, x)$ of the coalition at x is defined as

$$e(C, x) = u(C) - \sum_{i\in C} x_i \tag{1}$$

where x_i represented the payoff of agent$_i$ in coalition C. Nucleolus is a distribution method to minimize the maximum degree of dissatisfaction of coalition in cooperative games. Formally, the optimized process of the nucleolus is as follows:

$$\min_{x} \max_{C\subseteq N} e(C, x) \tag{2}$$

Since there are n agents, there are a total of 2^n coalitions (including empty sets), so for a set of imputation x, there are always 2^n excesses. And then sort excesses by non-increment called excesses sequence

$\theta(x)$, and it is defined as follows:

$$\theta(x) = [e(C_1, x), e(C_2, x), \ldots, e(C_{2^n}, x)] \tag{3}$$

The final nucleolus x^* is a unique payoff allocation that satisfies:

$$x^* = \{\theta(x) \preceq \theta(y) \mid \forall y \in I(v)\} \tag{4}$$

where $\preceq$ is lexicographical order and $I(v)$ is the set of all possible payoff distributions x, we call $\theta(x)$ as the Nucleolus. Note that the nucleolus always exists and lies in the core (if the core is non-empty), and it is unique [27].

Markov Core. In cooperative game theory, the Core [29] is the set of feasible allocations or imputations where no coalition of agents can benefit by breaking away from the grand coalition. For MDP, [33] extended the definition of the core in cooperative game theory, called Markov core. It can assess whether the payoff distribution during the learning process in an MDP meets certain conditions, specifically that the current payoff prevents all agents from having the incentive to leave the current coalition and form new coalitions to obtain greater rewards. Formally, we modify the Markov core (MC) for our defined EC-POMDP in Subsection 3.1 as follows:

$$MC = \left\{\left(\max_{a_i} x_i(s, a_i)\right)_{i\in N} \Big| \max_{a_C} x(s, a_C|C) \geq \max_{a_C} v(s, a_C), \forall C \subseteq N, \forall s \in S\right\} \tag{5}$$

where $x_i(s, a_i)$ represent the imputation of agent i by given state s and action a_i, $x(s, a_C|C) = \sum_{i\in C} x_i(s, a_i)$ is the imputation of coalition C by given state s and joint action $a_C \in A_C = \times_{i\in C} A_i$, and $v(s, a_C) \in \mathbb{R}_0^+$ is the assets of coalition C by given state s and coalition joint action a_C.

Markov Nucleolus. Based on the nucleolus in the cooperative game and the Markov core, we propose the Markov nucleolus to describe the credit assignment process in MARL. In EC-POMDP, $v(s, a_{CS})$ is the total assets under state s and joint action a while CS is the coalition structure. For convience, we shorten $v(s, a_{CS})$ as $v(s, a)$. $I(v(s, a))$ represents all possible payoff distributions from the total assets to the individual payoff and each $x(s, a) = (x_1(s, a_1), \ldots, x_n(s, a_n)) \in I(v(s, a))$ is one of payoff distribution. For any coalition C, the unity function is defined as $u(s, a|C) = \max_{a_C} v(s, a_C)$. So the excess under $x(s, a)$ is defined as follows:

$$\begin{aligned} e(C, x(s, a)) &= u(s, a|C) - x(s, a|C) \\ &= \max_{a_C} v(s, a_C) - \sum_{i\in C} x_i(s, a_i) \end{aligned} \tag{6}$$

Next, we sort the excesses of agents N with 2^n coalitions under $x(s, a)$ by the non-increment as excess sequence:

$$\theta(x(s, a)) = [e(C_1, x(s, a)), \ldots, e(C_{2^n}, x(s, a))] \tag{7}$$

Therefore, we can formally define the Markov Nucleolus for EC-POMDP as follows:

Definition 3.1. For EC-POMDP, the Markov Nuclelus $x^*(s, a) \in I(v(s, a))$ is the payoff distribution that satisfied with:

$$x^*(s, a) = \{\theta(x(s, a)) \preceq \theta(y(s, a)) \mid \forall y(s, a) \in I(v(s, a))\} \tag{8}$$

which is the smallest lexicographical order.

Note that the definition of the Markov nucleolus is a natural extension of the original concept from static cooperative games to MARL. We assume the Markov Nucleolus inherits the property that a nucleolus owns: there is always a unique solution, and if the Markov core is non-empty, the solution falls in the Markov core.

Nucleolus Q-value. We use the Markov nucleolus above to assign the global Q-value to individual Q-values. In EC-POMDP, the global Q-value under a coalition structure CS is defined as:

$$Q_{CS,global}(s, a) = R + \mathbb{E}\left[\max_{CS'} \sum_{C\in CS'} \sum_{i\in C} Q_i(s', a_i') \middle| \begin{array}{c} s' \sim P(\cdot|s, a) \\ a_i' = \max_{a_i} Q_i(s', a_i) \end{array}\right] \tag{9}$$

Next, we propose a Nucleolus Q-value based on the Markov nucleolus as follows:

Corollary 3.2. *To assign the global Q-value $Q_{CS,global}(s, a)$ by given state s and joint action a under coalition structure CS, we modify the payoff distribution as:*

$$Q_{CS}(s, a) = \left\{Q_{CS,1}(s, a_1), \ldots, Q_{CS,n}(s, a_n)\right\} \tag{10}$$

where $Q_{CS,i}(s, a_i)$ is the individual Q-value of agent i and $\sum_{i\in N} Q_{CS,i}(s, a_i) = Q_{CS,global}(s, a)$. Then, we model the Eq 6 to define the excess in coalition C in Q-learning.

$$e(C, Q_{CS}(s, a)) = \max_{a_C} V(s, a_C) - \sum_{i\in C} Q_{CS,i}(s, a_i) \tag{11}$$

where $V(s, a_C)$ is the Q-value of coalition C with given state s and joint action $a_C \in A = \times_{i\in C} A_i$ and $V(s, a_C) \in \mathbb{R}_0^+$. We use a similar definition (Eq 7) of the excess sequence in Markov nucleolus as:

$$\theta(Q_{CS}(s, a)) = [e(C_1, Q_{CS}(s, a)), e(C_{2^n}, Q_{CS}(s, a))] \tag{12}$$

Then the Nucleolus Q-value $Q^*_{CS}(s, a)$ can be formally defined as:

$$Q^*_{CS}(s, a) = \{\theta(Q^*_{CS}(s, a)) \preceq \theta(Q_{CS}(s, a)) | \forall Q_{CS}(s, a) \in I(Q_{CS}(s, a))\} \tag{13}$$

where $I(Q_{CS.global}(s, a))$ is all possible payoff distribution coalition structure CS.

Further, we suppose that $Q^*_{CS}(s, a) = w(s, a)Q(s, a)$, where $w(s, a)$ is a vector consisted as $[w_i(s, a)]_{i\in N}^\top$. Based on Bellman's optimality equation, we derive the Bellman optimality equation for the Markov nucleolus as follows:

$$\begin{aligned} Q^*_{CS}(s, a) =& w(s, a) \sum_{s'\in S} P(s'|s, a)[R + \gamma \max_{CS, a'} Q_{CS}(s', a')] \\ =& w(s, a) \sum_{s'\in S} P(s'|s, a)[R + \gamma \max_{CS, a'} \sum_{C\in CS} \sum_{i\in C} Q^*_{CS,i}(s', a_i')] \end{aligned} \tag{14}$$

Using the nucleolus-optimal Bellman equation, we derive the optimal nucleolus Q-value. To ensure the coalition's stability, we need to guarantee that the optimal nucleolus Q-value prevents agents from gaining a higher Q-value by leaving the current coalition to form a new one. And through Eq 14, we find that it is necessary to simultaneously optimize both the coalition structure and the action tuple $< CS, a >$ to find the optimal nucleolus Q-value. But this can result in exponential growth in the search space for the Q-values, which typically causes Q-learning to fail. To address these two issues, we derive Theorem 3.3 as follows:

Theorem 3.3. *The Nucleolus Q-value in EC-POMDP can guarantee*

(1) *Each agent's individual Q-value in the optimal coalition structure is greater than it is in other coalition structures;*

(2) *The actions and the coalition structure exhibit consistency, meaning that the coalition formed under the optimal actions is also optimal.*

The detailed proof for Theorem 1 is provided in **Appendix A.1**. Consequently, the optimal nucleolus-based Bellman equation can be reformulated as follows:

$$Q^*_{CS}(s,a) = w(s,a)\sum_{s'\in S} P(s'|s,a)[R + \gamma \max_{a'} \sum_{i\in N} Q^*_{CS',i}(s',a')] \tag{15}$$

Nucleolus Q Operator. We construct a constrained MARL Q operator to obtain the Nucleolus Q-value and provide a theorem showing that this operator can help us achieve the optimal Nucleolus Q-value. According to the definition of the Markov Nucleolus (Subsection 3.1), we need to minimize the maximum excess value. Inspired by Reward-Constrained Policy Optimization (RCPO) [32] and Discounted Reward Risk-Sensitive Actor-Critic [19], we introduce the maximum excess as a constraint $\xi(s,a)$ in the Q-learning process.

$$\xi(s,a) = \max_{a_C}[V(s,a_C) - \sum_{i\in C} Q^*_{CS,i}(s,a_i)] \tag{16}$$

Further, we construct the Lagrange multipliers as follows:

$$L(\lambda,a) = \min_{\lambda\geq 0}\max_{a}[\sum_{i\in CS} Q^*_{CS,i}(s,a_i) + \lambda\xi(s,a)] \tag{17}$$

According to RCPO, we need to implement a two-time-scale approach. This requires keeping λ constant on the faster time scale, and optimizing policy to fix the current policy on the slower time scale and optimize λ. This process allows us to identify the saddle point of Eq 17, which provides a feasible solution. Then, we propose an optimization operator, i.e., $\mathcal{H} : \times_{i\in N} Q^*_{CS,i}(s,a_i) \to \times_{i\in N} Q^*_{CS,i}(s,a_i)$, with nucleolus constraints as follows:

$$\begin{aligned}\mathcal{H}(\times_{i\in N} Q^*_i(s,a_i)) =& w(s,a)\sum_{s'\in S} P(s'|s,a)\\ & [R + \gamma \max_{a'} \sum_{i\in CS'} Q^*_{CS',i}(s,a'_i) + \lambda\xi(s,a)]\end{aligned} \tag{18}$$

Therefore, we demonstrate that using this new operator, Q learning can converge to the optimal Nucleolus Q-value by Theorem 3.4 below, where the detailed proof is provided in **Appendix A.2**.

Theorem 3.4. *Nucleolus-based Bellman operator can converge the optimal Nucleolus Q-value and the corresponding optimal joint deterministic policy when* $\sum_{i\in N}\max_{a_i} w_i(s,a_i) \leq \frac{1}{\gamma+\lambda}$

Consequently, we believe nucleolus Q learning can help us find the maximum social welfare based on nucleolus allocation. We can conclude that no agent has an incentive to deviate from the current coalition, and this allocation provides an explanation for the credit assignment of global rewards in cooperative MARL.

Implementation in Practice. We replace the global state s in individual Q network $Q_i(s,a_i)$ with the history of local observations $\tau_i \in \tau$ for agent i, resulting in $Q_i(\tau_i,a_i)$. We handle the sequence of local observations τ_i using a recurrent neural network (RNN) [30]. Thus, the proposed nucleolus Q-learning from Eq 15 can be rewritten as:

$$Q^*_{CS}(\tau,a) = w(s,a)\sum_{s'\in S} P(s'|s,a)[R + \max_{a'} \sum_{i\in N} Q^*_{CS',i}(\tau'_i,a'_i)] \tag{19}$$

In MARL, the coalition utility function is usually unknown and must be learned through exploration. We designed our utility function network using a multi-Layer perceptron network [24] based on the critic function network in RCPO. In RCPO, the temporal difference error of the utility function network updates is as follows.

$$\begin{aligned}\Delta_V(s,a,s') =& R + \lambda\left[\max_{a_C}[V(s,a_C;\phi^-) - \sum_{i\in C} Q^*_{CS,i}(s,a_i)]\right]\\ & + \eta_1 \max_{a'_C} \sum_{C\in CS'} V(s',a'_C;\phi^-) - \sum_{C\in CS} V(s,a_C;\phi)\end{aligned} \tag{20}$$

where η_1 is the discounting rate of the utility function network. Similar to RCPO, when updating the utility function network, we treat the constraints $\xi(s,a)$ as part of the reward. This indicates the constraint term involving the utility network is not updated, so we use the target network $V(\cdot;\phi^-)$ instead.

Unlike the update of the utility function network, the update of the Q network is performed using the Lagrange multiplier method. This implies that we need to optimize the Q network within the constraint conditions. Based on Eq 17, the update of network parameters ω, including both the individual Q network parameters φ and the hypernetwork parameters ψ shows as follows:

$$\begin{aligned}\nabla_\omega L(\lambda,\omega) =& \min_{\omega} \mathbb{E}\Bigg[R + \lambda\big[\max_{a_C}[V(s,a_C) - \sum_{i\in C} Q^*_{CS,i}(\tau_i,a_i;\varphi_i)]\big]\\ & + \gamma \sum_{i\in CS'} \max_{a'}[w(s',a';\psi^-) Q^*_{CS'}(\tau',a';\varphi_i^-)]\\ & - \sum_{i\in CS} w(s,a;\psi) Q^*_{CS}(\tau,a;\varphi_i)\Bigg]\end{aligned} \tag{21}$$

where φ^- represents the target Q network parameters, and ψ^- represents the target hypernetwork parameters. The update of multiplier λ is shown as follows:

$$\nabla_\lambda L(\lambda,\omega) = \mathbb{E}[\lambda - \eta_2\xi(s,a)] \tag{22}$$

where η_2 is the learning rate of the update of λ.

For any coalition C, the input to the unity network consists of the current global state s and the actions of all agents within the coalition. The network's output is the estimated value for this state-action pair. The input actions are provided by the individual Q network of each agent. To ensure a consistent input length, we use agent IDs for position encoding, placing each agent's action in the corresponding position based on its ID. For agents that are not part of the coalition C, we use placeholders to fill in the missing action inputs, maintaining a uniform input structure in different coalitions. In addition, we present a pseudo-code for our proposed nucleolus-based credit assignment as Algorithm 1, which provides a step-by-step flow of the learning process.

4 EXPERIMENTAL SETUP

We validate our algorithm on two popular MARL benchmarks: Predator-Prey [3] and the StarCraft Multi-Agent Challenge (SMAC)

Algorithm 1 Nucleolus-based Credit Assignment Algorithm

```
1: Initialize parameter $\varphi$ for agent Q network and target parameter $\varphi^-$ by copying parameter $\varphi$;
2: Initialize parameter $\phi$ for coalition unity network and target parameter $\phi^-$ by copying parameter $\phi$ ;
3: Initialize multiplier $\lambda \geq 0$, learning rate $\eta_1 > \eta_2 > \eta_3$ and discounting rate $\gamma$;
4: Initialize replay buffer $\mathcal{B}$;
5: while $Q(o_i, a_i; \omega_i)$ network not converge do
6:     for Time $t = 1, 2, \ldots to T$ do
7:         Observe a global state $s$ and agent's observation $o_i$;
8:         Select action $a_i$ according to each agent's Q network;
9:         Execute action $a$ and get next state $s'$, each agent's next observation $o_i'$ and reward $R$;
10:        Store $< s, o_i \in o, s', a, R >$ to $\mathcal{B}$;
11:    end for
12:    Sample K episodes from $\mathcal{B}$;
13:    for Episode $k = 1, 2, \ldots to D$ do
14:        for Time $t = 1, 2, ..., T$ do
15:            Get next time step action $a'$ by $\max_{a'} Q(s', a')$;
16:            Coalition unity network update:
                   $\phi_{t+1} \leftarrow \phi_t + \eta_1 \nabla_{\phi_t} [R + \lambda \xi(s, a; \phi^-) + \gamma V(s', a'; \phi^-) - V(s, a; \phi_t)]$;   ▷ Eq 20
17:            Nucleolus Q network update:
                   $\varphi_{t+1} \leftarrow \varphi_t + \eta_2 \nabla_{\varphi_t} [R + \lambda \xi(s, a; \varphi_t) + \gamma Q(s', a'; \varphi^-) - Q(s, a; \varphi_t)]$;   ▷ Eq 21
18:        end for
19:        Lagrange multiplier update:
               $\lambda \leftarrow \lambda - \eta_3 \xi(s, a)$;   ▷ Eq 22
20:        Update all target parameters by copying;
21:    end for
22: end while
```

[26], against four well-known baselines, including VDN [31], QMIX [23], WQMIX [22], and SHAQ [33]. All baseline implementations are derived from PyMARL2 [12].

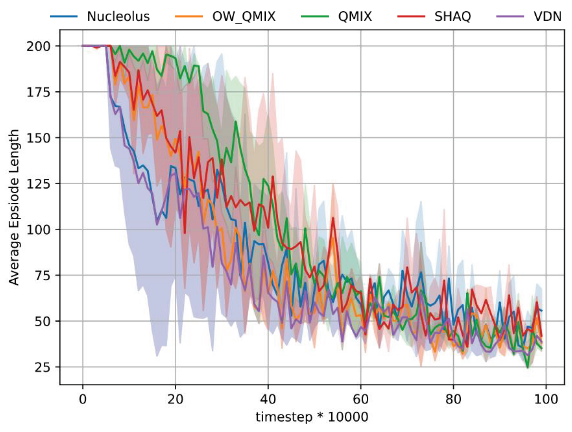


Figure 2: Learning performance in Predator-Prey: the turns to catch prey on the test episode

Predator-Prey. The Predator-Prey environment is a testbed simulating the interaction between predators and prey. Predators must cooperate to capture prey, while the prey attempts to evade capture and maximize their survival time. Each agent operates based on local observations, with predators compensating for their individual limitations through cooperation and prey using strategic movement to escape. Predators are rewarded for successful captures, which require at least two predators to capture a prey simultaneously. If the prey is captured, a global reward of 10 points is granted. The primary mission for each predator is to minimize the number of steps required to capture prey by coordinating effectively. The state space for each predator includes its position, velocity, the relative displacement between itself and the prey or other predators, and the velocity of the prey. We set 8 Predators and 4 Prey in this task, and the training duration in the Predator-Prey environment is set to 1 million steps for all algorithms.

StarCraft Multi-Agent Challenge. SMAC is a widely used benchmark for evaluating MARL algorithms. In SMAC, each unit is controlled by an RL agent, which dynamically adapts its behavior based on local observations. Each agent is tasked with learning cooperative strategies to optimize performance in combat scenarios. Agents have a limited line of sight to perceive their local environment and can only attack enemy units within their attack range. The objective is to maximize cumulative rewards, which are calculated based on the damage inflicted on enemy units and the number of enemy units destroyed, with substantial rewards awarded for achieving overall victory. In our experiment, the training duration varies from 500k to 2.5 million steps, depending on the task difficulty. Algorithm performance is evaluated every 10k steps across 32 episodes. Detailed information on all hyperparameters of the network can be found in **Appendix B**, and descriptions of each task are present in **Appendix C**. The source code of our proposed algorithm can be reviewed in **Appendix D**, and the details of the computing platform are described in **Appendix E**.

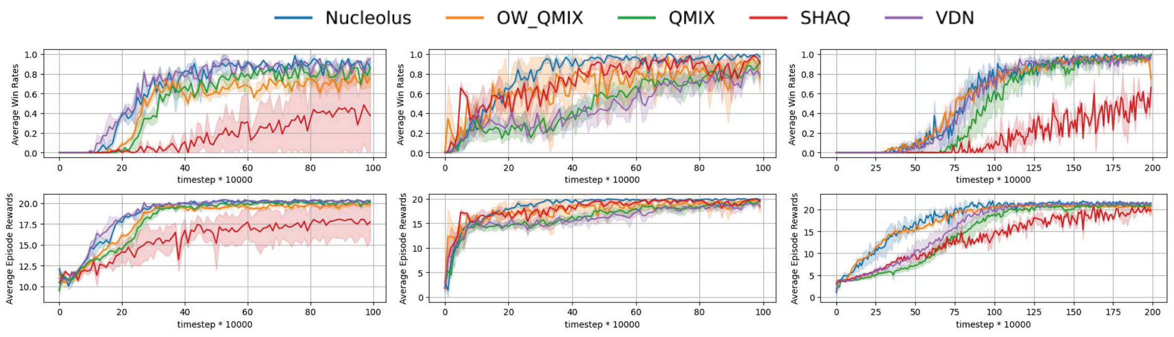
5 RESULTS AND ANALYSIS

In this section, we present the performance of our algorithm compared to baseline methods across two different environments: Predator-Prey and SMAC. We further analyze the coalition formation process across several tasks, highlighting how our method improves agents' cooperation in MARL.

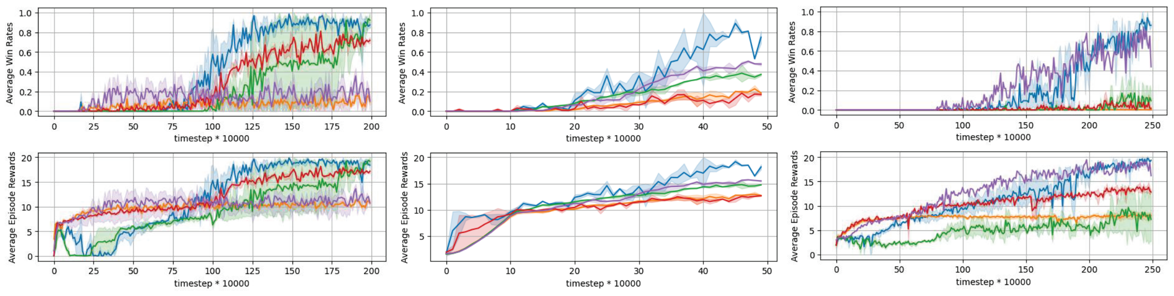
5.1 Learning Performance

Predator-Prey. **Fig. 2** shows the comparative performance of the five algorithms in the Predator-Prey environment, focusing on convergence speed, stable performance, and variance. VDN achieves quick convergence with low variance, yielding stable outcomes, but its overall performance is moderate. QMIX, while slower to converge, demonstrates the best final performance after 95,000 steps, capturing the prey in approximately 25 turns, though with greater fluctuations. SHAQ also exhibits slower convergence and larger performance variances.

In contrast, our proposed algorithm (nucleolus) combines rapid convergence with highly efficient capture strategies, significantly reducing the number of steps needed for successful captures. While it shows relatively higher variance, the speed and effectiveness



(a) 2s3z (b) 2c vs 64zg (c) 3s vs 5z



(d) 5m vs 6m (e) 8m vs 9m (f) corridor

Figure 3: Learning performance in SMAC: median test win and rewards for easy task (a), hard (b-e) and super-hard (f) maps.

in learning make it particularly advantageous for time-sensitive tasks. This positions our method as a strong contender, especially in scenarios demanding fast adaptation and decision-making.

SMAC. The training curves for all algorithms in SMAC are shown in **Fig. 3**, indicating varied performance across six tasks in various difficulty levels (easy, hard and super-hard).

2s3z: Our algorithm (nucleolus) gradually increases, reaching around 80% after approximately 40,000 timesteps, and eventually approaching 100%, showing stable and fast convergence. OW-QMIX behaves similarly to ours, ultimately also approaching 100% but slightly lower than ours. QMIX performs slightly worse, with the final win rate stabilizing at around 80%. SHAQ has a lower win rate throughout the training, stabilizing at around 45% with high variance.VDN performs well, with a final win rate of about 85%, though with slight fluctuations.

2c vs 64zg: Our algorithm (nucleolus) rises rapidly early on, reaching a win rate of over 90% after about 40,000 timesteps and stabilizing near 100%. OW-QMIX and SHAQ have a similar convergence speed to ours, reaching around an 80% win rate at 40,000 timesteps and stabilizing at about 90%, slightly lower than ours. QMIX and VDN converge the slowest, with a final win rate stabilizing around 80%.

3s vs 5z: Our algorithm (nucleolus), OW-QMIX, and VDN perform well, with win rates rapidly increasing after around 300,000 timesteps and reaching approximately 80% at 1,000,000 timesteps. The final win rate is close to 100%. QMIX lags behind the other three algorithms, with a win rate reaching around 60% after 1,000,000 timesteps, and eventually approaching 100%. SHAQ converges the slowest, with the final win rate not exceeding 60%.

5m vs 6m: Our algorithm (nucleolus) converges the fastest, with the win rate rapidly rising after around 100,000 timesteps and eventually approaching 100%. OW-QMIX and VDN never exceed a win rate of 40% throughout the training period. QMIX converges slowly and with high variance but eventually reaches a win rate close to 100%. SHAQ converges second fastest after ours, with a final win rate stabilizing around 70%.

8m vs 9m: Our algorithm (nucleolus) has the fastest convergence, with the win rate increasing rapidly early on, significantly improving after around 30,000 timesteps, and ultimately approaching 80%. VDN converges more slowly but improves after 30,000 timesteps, with a final win rate of about 50%. QMIX is relatively stable but increases more slowly, with the final win rate stabilizing around 40%. SHAQ and OW-QMIX perform worse, with slow win rate improvements, eventually reaching around 20% at 50,000 timesteps.

Corridor: Our algorithm (nucleolus) converges relatively quickly, with a final win rate exceeding 90%, making it the best-performing algorithm. VDN has the fastest convergence, with a final win rate close to 80%. QMIX, SHAQ, and OW-QMIX all have win rates that never exceed 20% throughout the training period.

Based on the performance across all tasks above, our algorithm consistently achieved the highest win rates compared to the other baseline algorithms, with a superior convergence speed. Notably, in the **5m vs 6m**, **8m vs 9m**, and **corridor** tasks, our algorithm outperformed the others in terms of final win rates. We attribute

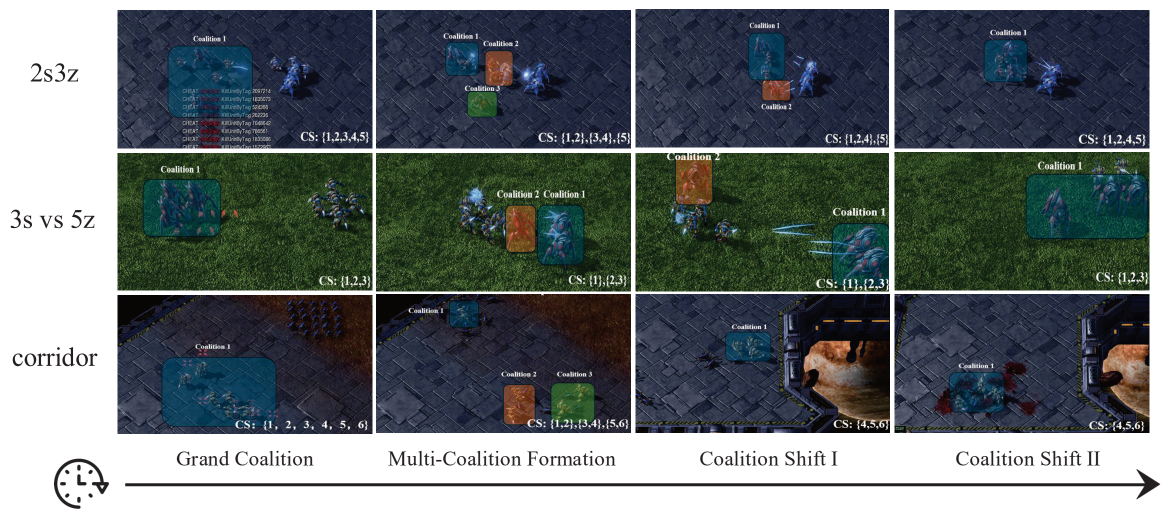


Figure 4: Visualize multiple coalition formation process in three tasks: 2s3z (easy), 3s vs 5z (hard) and corridor (super-hard)

these tasks are asymmetric, where the number of enemy agents significantly exceeds the number of our agents, and we found more difficult tasks require our agents to form small, dispersed alliances to succeed. In contrast, in the **2c vs 64zg** task, although the enemy agents still outnumber our agents, the smaller number of our agents makes it difficult to form effective coalitions, which limits the performance advantage compared to other algorithms. This enables our algorithm to potentially excel in handling more challenging tasks. As the number of agents increases, it can form more small coalitions, which may lead to a significantly improved performance

5.2 Multiple Coalition Formations

To further validate the effectiveness of our coalition formation in MARL, we visualized the coalition process on three challenging SMAC maps: 2s3z, 3svs 5z, and corridor. **Fig. 4** illustrates how multiple coalitions are formed and adapted to complete a composite task.

***2s3z*.** Initially, all agents belong to the *Grand Coalition*. In *Multi-Coalition Formation*, agents form multi-coalition, Coalition 2 (2 Zealots) draws fire on the 2 enemy Stalkers, while Coalition 1 (1 Stalker and 1 Zealot) takes the opportunity to attack the enemy Zealots, and Coalition 3 (1 Stalker) repositions, looking for a target. In *Coalition Shift I*, agents shift coalition structure and form new coalitions, the remaining Stalker in Coalition 2 attacks the enemy Stalkers, while the remaining units in Coalition 1 focus fire on the enemy units other than the Stalkers. Eventually, in *Coalition Shift II*, the remaining agents reform into a grand coalition to focus fire on the remaining enemy Stalkers.

***3s vs 5z*.** At the start, all agents belong to the *Grand Coalition*. In *Multi-Coalition Formation*, agents form multi-coalition, Coalition 2 (1 Stalker) draws all enemy fire, while Coalition 1 (2 Stalkers) takes the opportunity to attack and eliminate one enemy Zealot. In *Coalition Shift I*, agents shift coalition structure and form new coalitions, the strategy is repeated, with Coalition 2 (1 Stalker) drawing enemy fire, and Coalition 1 (2 Stalkers) focusing on eliminating the remaining enemy units. As a final point, in *Coalition Shift II*, the remaining agents reform into a grand coalition to focus fire on the remaining enemy Zealots.

***Corridor*.** At the beginning, all agents belong to the *Grand Coalition*. In *Multi-Coalition Formation*, agents form multi-coalition, Coalition 1 (2 Zealots) draws most of the enemy fire to create space for the main force to reposition, while Coalition 2 (2 Zealots) and Coalition 3 (2 Zealots) form a local numerical advantage to focus fire and eliminate smaller enemy units. In *Coalition Shift I*, agents shift coalition structure and form new coalitions, the remaining agents form a grand coalition to gradually eliminate the remaining enemy units. Ultimately, in *Coalition Shift II*, this strategy is repeated until all enemy units are eliminated.

6 CONCLUSION

We introduced a nucleolus-based credit assignment method to create multiple effective coalitions for cooperative MARL, leveraging cooperative game theory to enable agents to form smaller, efficient coalitions to tackle complex tasks. Unlike traditional methods that rely on a single grand coalition, our method could ensure fair and stable credit distribution across multiple coalitions. Experiments in Predator-Prey and StarCraft scenarios showed that our method consistently outperformed baseline approaches in learning speed, cumulative rewards, and win rates, demonstrating improvements in training efficiency, task performance, interpretability, and stability. Future research would focus on scaling nucleolus MARL to larger multi-agent environments, considering the computational efficiency of nucleolus calculations.

ACKNOWLEDGMENTS

This study is supported by Australian Research Council (ARC) DECRA Fellowship DE220100265 and ARC Discovery Project DP250103612.

REFERENCES

[1] Haris Aziz, Bo Li, Shiji Xing, and Yu Zhou. 2023. Possible Fairness for Allocating Indivisible Resources. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*. 197–205.

[2] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2018. Emergent Complexity via Multi-Agent Competition. In *International Conference on Learning Representations*.

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

[4] Rodica Branzei, Dinko Dimitrov, and Stef Tijs. 2008. *Models in cooperative game theory*. Vol. 556.

[5] Trevor Campbell, Luke Johnson, and Jonathan P How. 2013. Multiagent allocation of markov decision process tasks. In *2013 American Control Conference*. IEEE, 2356–2361.

[6] Wubing Chen, Wenbin Li, Xiao Liu, Shangdong Yang, and Yang Gao. 2023. Learning explicit credit assignment for cooperative multi-agent reinforcement learning via polarization policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11542–11550.

[7] Alessandro Farinelli, Antonello Contini, Davide Zorzi, et al. 2020. Decentralized Task Assignment for Multi-item Pickup and Delivery in Logistic Scenarios.. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*. 1843–1845.

[8] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[9] Mukesh Gautam and Mohammed Benidris. 2023. Coalitional Game Theory in Power Systems: Applications, Challenges, and Future Directions. In *2023 IEEE Texas Power and Energy Conference*. 1–6.

[10] Sven Gronauer and Klaus Diepold. 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review* 55, 2 (2022), 895–943.

[11] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. 2021. Explainability in deep reinforcement learning. *Knowledge-Based Systems* 214 (2021), 106685.

[12] Jian Hu, Siying Wang, Siyang Jiang, and Musk Wang. 2023. Rethinking the Implementation Tricks and Monotonicity Constraint in Cooperative Multi-agent Reinforcement Learning. In *The Second Blogpost Track at The International Conference on Learning Representations*.

[13] Shariq Iqbal, Robby Costales, and Fei Sha. 2022. Alma: Hierarchical learning for composite multi-agent tasks. In *Advances in Neural Information Processing Systems*, Vol. 35. 7155–7166.

[14] Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. 2021. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*. 4596–4606.

[15] Seung Hyun Kim, Neale Van Stralen, Girish Chowdhary, and Huy T. Tran. 2022. Disentangling Successor Features for Coordination in Multi-agent Reinforcement Learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 751–760.

[16] Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Anima Anandkumar. 2021. Coach-player multi-agent reinforcement learning for dynamic team composition. In *International Conference on Machine Learning*. 6860–6870.

[17] Frans A Oliehoek. 2012. Decentralized pomdps. *Reinforcement learning: state-of-the-art* (2012), 471–503.

[18] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[19] LA Prashanth and Mohammad Ghavamzadeh. 2016. Variance-constrained actor-critic algorithms for discounted and average reward MDPs. *Machine Learning* 105 (2016), 367–417.

[20] Scott Proper and Prasad Tadepalli. 2009. Solving multiagent assignment markov decision processes. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. 681–688.

[21] Anatol Rapoport. 2012. *Game theory as a theory of conflict resolution*. Vol. 2.

[22] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 33. 10199–10210.

[23] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* 21, 178 (2020), 1–51.

[24] Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65, 6 (1958), 386.

[25] Franco Ruggeri, William Emanuelsson, Ahmad Terra, Rafia Inam, and Karl H. Johansson. 2024. Rollout-based Shapley Values for Explainable Cooperative Multi-Agent Reinforcement Learning. In *2024 IEEE International Conference on Machine Learning for Communication and Networking*. 227–233.

[26] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2186–2188.

[27] David Schmeidler. 1969. The nucleolus of a characteristic function game. *SIAM journal on applied mathematics* 17, 6 (1969), 1163–1170.

[28] Lloyd S Shapley. 1953. A value for n-person games. *Contribution to the Theory of Games* 2 (1953).

[29] Lloyd S Shapley and Martin Shubik. 1971. The assignment game I: The core. *International journal of game theory* 1, 1 (1971), 111–130.

[30] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306.

[31] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius 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*. 2085–2087.

[32] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. 2019. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*.

[33] Jianhong Wang, Yuan Zhang, Yunjie Gu, and Tae-Kyun Kim. 2022. Shaq: Incorporating shapley value theory into multi-agent q-learning. In *Advances in Neural Information Processing Systems*, Vol. 35. 5941–5954.

[34] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020. Shapley Q-value: A local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7285–7292.

[35] Nihat Öner and Gültekin Kuyzu. 2022. Nucleolus based cost allocation methods for a class of constrained lane covering games. *Computers and Industrial Engineering* 172 (2022), 108583.