Multi-Agent Synchronization Tasks

Rolando Fernandez
DEVCOM Army Research Laboratory
The University of Texas at Austin
Austin, TX, United States
rfernandez@utexas.edu

Garrett Warnell
DEVCOM Army Research Laboratory
The University of Texas at Austin
Austin, TX, United States
garrett.a.warnell.civ@army.mil

Derrik E. Asher
DEVCOM Army Research Laboratory
Adelphi, MD, United States
derrik.e.asher.civ@army.mil

Peter Stone
The University of Texas at Austin
Sony AI
Austin, TX, United States
pstone@cs.utexas.edu

ABSTRACT

In multi-agent reinforcement learning (MARL), coordination plays a crucial role in enhancing agents' performance beyond what they could achieve through cooperation alone. The interdependence of agents' actions, coupled with the need for communication, leads to a domain where effective coordination is crucial. In this paper, we introduce and define *Multi-Agent Synchronization Tasks* (MSTs), a novel subset of multi-agent tasks. We describe one MST, that we call *Synchronized Predator-Prey*, offering a detailed description that will serve as the basis for evaluating a selection of recent state-of-the-art (SOTA) MARL algorithms explicitly designed to address coordination challenges through the use of communication strategies. Furthermore, we present empirical evidence that reveals the limitations of the algorithms assessed to solve MSTs, demonstrating their inability to scale effectively beyond 2-agent coordination tasks in scenarios where communication is a requisite component. Finally, the results raise questions about the applicability of recent SOTA approaches for complex coordination tasks (i.e. MSTs) and prompt further exploration into the underlying causes of their limitations in this context.

KEYWORDS

Coordination, Multi-Agent Reinforcement Learning, Graph Neural Networks, Coordination Graphs, Multi-Agent Synchronization Tasks, Predator-Prey

1 INTRODUCTION

Coordination is essential in fully cooperative environments where agents on a team must work together to achieve a common goal. In this work, we define *Coordination* as the ability of agents to align their actions with their partners and *Cooperation* as the ability of agents to work together to achieve a shared objective (e.g., by sharing information about their actions to gain mutual benefit) [1, 4]. Together *Coordination* and *Cooperation* allow agents to communicate and collaborate to make joint decisions and take collective actions that lead to the best outcomes for the team.

In multi-agent reinforcement learning (MARL), coordination plays a crucial role in enhancing agents' performance beyond what they could achieve through cooperation alone. This performance improvement is achieved by enabling agents to share information and align their actions. Coordination is essential in MARL because it enables teams to work together more effectively than without coordination, especially when the agents' objectives are interdependent or when their actions can impact each other. For example, in a soccer game, players must coordinate their actions to score and defend their goals.

Typically, MARL approaches leverage cooperative domains tailored to facilitate the study of agent interactions and coordination strategies. These cooperative environments share common attributes essential for testing and refining multi-agent systems. Cooperative domains are characterized by agents working toward shared goals, where the success of each agent is intertwined with the overall team objective. The interdependence of agents' actions, coupled with the need for communication, leads to a domain where effective coordination is crucial. Tasks within these domains are deliberately complex, often featuring dynamic elements and uncertainties that demand adaptive coordination strategies. Some examples of these types of domains are predator-prey, stag hunt, and StarCraft, among others [2, 8–10, 14]. Additionally, one-shot game domains have been employed to analyze various MARL methods, contributing to a comprehensive understanding of the capabilities and limitations of the methods [3, 8]. These environments provide diverse challenges for evaluating the robustness and adaptability of MARL approaches.

In this paper, the focus is directed towards Multi-agent Synchronization Tasks (MSTs), which are a novel subset of cooperative multi-agent domains. MSTs necessitate communication strategies and precise timing for agents to align their actions and accomplish a given task. The emphasis on communication underscores the importance of deliberate information exchange among agents, highlighting the need for a well-defined communication framework to achieve coordination. The requirement for precise timing further accentuates the intricacies of these domains, where the synchronization of agents' actions becomes a critical factor in the successful execution of tasks. By honing in on MSTs, the paper aims to contribute insights into the challenges and strategies associated with explicit communication and precise timing in multi-agent systems, thereby advancing our understanding of cooperative behaviors in complex environments.

We will describe one such domain, that we call *Synchronized Predator-Prey*, offering a detailed description that will serve as the

Proc. of the Adaptive and Learning Agents Workshop (ALA 2024), Avalos, Müller, Wang, Yates (eds.), May 6-7, 2024, Online, https://ala2024.github.io/. 2024.

basis for evaluating a selection of recent MARL algorithms explicitly designed to address coordination challenges through the use of communication strategies. Our analysis will shed light on how the existence of penalties for mis-timed (unsychrnonized) actions, manifested as negative rewards, influences the need for communication between agents (i.e. to avoid relative overgeneralization [13]). Furthermore, we will present empirical evidence that reveals the limitations of the assessed MARL algorithms, demonstrating their inability to scale effectively beyond two-agent coordination tasks in scenarios where communication is a requisite component. This investigation aims to provide an understanding of the interplay between communication, penalties, and scalability within cooperative multi-agent systems, offering valuable insights into the performance and limitations of contemporary MARL approaches in such contexts.

2 RELATED WORK

We now discuss the relationship between Multi-agent Synchronization Tasks (MSTs) and the MARL literature. We will discuss multi-agent synchronization tasks in the context of several well-known MARL domains. Additionally, we will discuss several current state-of-the-art (SOTA) MARL algorithms that we will evaluate on synchronization tasks.

2.1 Domains

In this work, we are concerned with a particular type of multi-agent problem that requires agents to align their actions to succeed in a given task. Such tasks have been explored in the MARL literature via established domains such as Stag Hunt and Predator-Prey Pursuit games [11, 15]. More recent approaches have also employed the StarCraft Multi-Agent Challenge (SMAC) domain and one-shot (i.e., non-sequential) domains [3, 8]. However, none of these domains highlights the need for precise synchronization among agents (see Section 3), and so we will introduce here a new domain that we refer to as *Synchronized Predator-Prey.*

2.2 Algorithms

We now briefly describe existing algorithmic approaches designed to engender coordination among agents in MARL. For more detailed information, refer to the methods/algorithms sections of the cited works. We focus on the use of communication methods to solve multi-agent synchronization tasks; specifically on the communication method called *Coordination Graphs* (CG), a graph representation of the communication structure between agents in a multi-agent system. While CGs were first combined with RL in the work of Guestrin et al. [5], there have been several recent advances in this space.

DCG. Building on the work of Guestrin, Bohmer et al. [2] proposed Deep Coordination Graphs (DCG), a MARL algorithm based on Deep Q Networks (DQN) that utilized CGs to coordinate greedy action selection between connected agents in a graph. Instead of the Variable Elimination scheme, they use the Max-Plus anytime messaging scheme [7], which trades the exact optimal joint action for an approximation that allows real-time action selection. Though this approach overcomes the real-time limitations of previous works, it still maintains the use of pre-defined coordination graphs with fixed topologies.

DICG. To overcome the limitations of using pre-defined fixed coordination graphs, Li et al. [9] proposed Deep Implicit Coordination Graphs (DICG). This method is a policy-based algorithm that allows implicit coordination graphs to be constructed implicitly at each timestep given the state of the environment. This new algorithm introduced two main changes to the previous work: (1) a Graph Neural Network (GNN) [6] is used to handle message passing between agents, and (2) an attention network is introduced, which implicitly creates the coordination graph at each timestep.

QGNN. Similar to the DICG application of GNNs, Kortvelesy et al. [8] proposed a DQN-based value function factorization method called QGNN that uses a model and mixer framework akin to QMIX [14], but with a GNN [6] embedded in the model network. To summarize, QGNN introduces three main changes to the previous work: (1) it utilizes a model and mixer framework like QMIX, (2) a GNN [6] is embedded in the model network, and (3) the QGNN mixer is introduced to compute the monotonic joint-action value using a custom pooling operation.

3 MULTI-AGENT SYNCHRONIZATION TASKS

In this paper, we are concerned with a particular subset of multi-agent tasks that we call multi-agent synchronization tasks (MSTs). In this section, we provide a precise technical definition of MSTs, and also discuss a sample domain that we have designed that serves as an exemplar.

3.1 Background

We define MSTs as a subset of decentralized partially observable Markov decision processes (Dec-POMDPs) [12]. Formally, for n agents, a Dec-POMDP is characterized by a tuple:

$$\mathcal{M}_{DecP} = \langle \mathbb{D}, \mathbb{S}, \mathbb{A}, T, \mathbb{O}, \sigma, R, \gamma \rangle.$$

Here, $\mathbb{D} = 1, \ldots, n$ is the set of n agents and $\mathbb{S}$ denotes a set of states. $\mathbb{A}$ is the set of joint actions $\mathbb{A} = \mathbb{A}^1 \times \ldots \times \mathbb{A}^n$, where $\mathbb{A}^i$ is the set of actions available to agent $i \in \mathbb{D}$, which can be different for each agent. At each timestep t, the system is in state $s_t \in \mathbb{S}$, and each agent i takes an action a_t^i, leading to one joint action $a_t = (a_t^1, \ldots, a_t^n)$ at each timestep. The resulting next state, s_{t+1}, is drawn from the transition function $T(s_{t+1} \mid s_t, a_t)$. The transition yields a shared reward $R_t = R(s_t, a_t)$ with $\gamma \in [0, 1)$ denoting the discount factor. Similar to $\mathbb{A}$, $\mathbb{O} = \mathbb{O}^1 \times \ldots \times \mathbb{O}^n$ is the set of joint observations, where $\mathbb{O}^i$ is the set of observations available to agent $i \in \mathbb{D}$. Each agent i observes the state only through a (partial) observation $o_t^i \in \mathbb{O}^i$ drawn from its observation function $\sigma^i(o \mid a_t, s_{t+1})$.

A solution to a Dec-POMDP is typically an optimal policy or a set of optimal policies that specify the actions each agent should execute, considering their individual histories of observations and actions, as informed by the optimal state-action value function (Q^*), where

$$Q^*(b, a) = \sum_{s \in \mathbb{S}} b(s) \left[R(s, a) + \gamma \sum_{o \in \mathbb{O}} \sigma(o|s, a) \max_{a'} Q^*(b', a') \right].$$

Because the state of the environment is not fully observable to the agents, these policies must be based on beliefs or estimates of the state (b), derived from the history of observations and actions. The overarching aim of these policies is to maximize the collective reward over time, optimizing the expected benefit for all agents involved across a potentially infinite or finite horizon. By striving to maximize this collective reward, the solution inherently seeks the most advantageous outcome for the group as a whole, navigating the complexities of partial observability and decentralized decision-making in Dec-POMDP environments.

3.2 MST Definition

Broadly speaking, MSTs are tasks that exhibit high stakes when it comes to agents synchronizing their actions. We characterize them using per-agent notions of *neutral actions* ($\mathbb{A}^{i,\, neut}$) and *synchronization actions* ($\mathbb{A}^{i,\, sync}$). Neutral actions are standard actions that an agent can take without having to coordinate with others, whereas synchronization actions require precise timing and coordination with other agents to be successful. Intuitively, synchronization actions are special actions that, in at least one circumstance, are optimal, but also carry the risk that, if attempted by some members of the team but not executed by others at the same time, the result is worse than if all members of the team had restricted themselves to neutral actions only.

The formal structure of these tasks involves dividing each agent's possible actions into synchronization and neutral actions, with the total action space for an agent being the union of these two types. This division allows us to categorize joint actions (actions taken by all agents together) into three subsets based on their outcomes in a given state, *neutral* joint actions ($\mathbb{A}^{neut}$), *synchronization-positive* joint actions ($\mathbb{A}^{+}(s)$), and *synchronization-negative* joint actions ($\mathbb{A}^{-}(s)$). $\mathbb{A}^{neut}$ consists of joint actions in which every agent opts for a neutral action. These actions are safe, but may not always lead to the best possible outcomes. $\mathbb{A}^{+}(s)$ is the set of actions that, when executed by the agents, result in a successful synchronization, offering a better reward than any set of neutral actions in the same state. ($\mathbb{A}^{-}(s)$) is the set of actions where attempts at synchronization fail, leading to outcomes worse than if all agents had chosen neutral actions. For a task to satisfy the definition of an MST, there must exist at least one state such that the sets $\mathbb{A}^{+}(s)$ and $\mathbb{A}^{-}(s)$ are not empty.

Moving a large/heavy piece of furniture is an example of an MST where two or more people must coordinate their actions to achieve task success (e.g., avoid hurting themselves and/or damaging the furniture). In this case, the $\mathbb{A}^{i,\, neut}$ actions consist of not lifting the piece of furniture and the $\mathbb{A}^{i,\, sync}$ actions consist of lifting the piece of furniture. The joint actions in $\mathbb{A}^{+}$ are those such that the two people lift up the piece of furniture simultaneously. Then, the joint actions in $\mathbb{A}^{-}$ are those such that one person lifts, while the other does not, or such that they both lift, but one person is not next to the furniture, as the lifter could get hurt, and/or the furniture could be damaged. While $\mathbb{A}^{neut}$, are those actions such that no person attempts to lift the piece of furniture, as no one can be injured and the furniture cannot be damaged. From the example, it is clear that actions resulting in successful synchronization are a better choice over neutral actions, while actions that result in unsuccessful synchronization are clearly worse.

More formally, let $(\mathbb{A}^{1,\, sync}, \ldots, \mathbb{A}^{n,\, sync})$ define a partitioning of each agent's action space into $\mathbb{A}^{i} = \mathbb{A}^{i,\, sync} \cup \mathbb{A}^{i,\, neut}$ where $\mathbb{A}^{i,\, neut} = \overline{\mathbb{A}^{i,\, sync}}$. Further, let that partitioning define the following three subsets of the joint action space:

$$Q^{*} = \mathbb{A}^{1,\, neut} \times \cdots \times \mathbb{A}^{n,\, neut},$$

$$\mathbb{A}^{+}(s) = \left\{ a \;\middle|\; \begin{array}{l} \exists\, i, j \text{ s.t. } \begin{array}{l} a^{i} \in \mathbb{A}^{i,\, sync} \\ a^{j} \in \mathbb{A}^{j,\, sync} \end{array} \\ \forall\, a^{neut} \in \mathbb{A}^{neut}\; Q^{*}(s, a) > Q^{*}(s, a^{neut}) \end{array} \right\},$$

and

$$\mathbb{A}^{-}(s) = \left\{ a \;\middle|\; \begin{array}{l} \exists\, i \text{ s.t. } a^{i} \in \mathbb{A}^{i,\, sync} \\ \forall\, a^{neut} \in \mathbb{A}^{neut}\; Q^{*}(s, a^{neut}) > Q^{*}(s, a) \end{array} \right\}.$$

Note that the definition of $\mathbb{A}^{+}(s)$ depends on *at least* two agents synchronizing, but does not exclude joint actions where more than two agents synchronize.

Definition 1. ***Multi-agent Synchronization Task.*** *A multi-agent synchronization task (MST) is a Dec-POMDP where there exists a partitioning* $(\mathbb{A}^{1,\, sync}, \ldots, \mathbb{A}^{n,\, sync})$ *and at least one* $s \in \mathbb{S}$ *such that* $\mathbb{A}^{+}(s) \neq \emptyset$ *and* $\mathbb{A}^{-}(s) \neq \emptyset$.

The presence of elements in the set $\mathbb{A}^{+}(s)$ indicates that, when the system is in state s, there is at least one sub-team of agents for which the execution of synchronization actions ($\mathbb{A}^{i,\, sync}$) is the optimal action. Conversely, non-empty $\mathbb{A}^{-}(s)$ indicates that, for the same state, there are scenarios where at least one agent attempting a synchronization action results in a catastrophic failure in the sense that, if all agents had stuck to neutral actions, the outcome would have been better. Both sets being non-empty for the same s creates the condition for a high-stakes scenario: it is possible for agents to either achieve the highest level of success through the correct application of synchronization actions, but it is also possible that trying to execute a synchronization action can lead to a worse outcome than trying no synchronization actions at all. This duality underlines the critical importance of precise coordination and timing among agents in MSTs.

3.3 Example MST: Synchronized Predator-Prey

We now describe *Synchronized Predator-Prey*, a new variant of the classical Predator-Prey domain that satisfies the definition of an MST.

States. The *Synchronized Predator-Prey* environment is a discrete 10 x 10 grid with 8 predators and 8 prey. However, when considering sub-teams of three predators, the number of predators increases to 9. An episode is initialized with all agents randomly placed on the game grid. An episode ends once all predator sub-teams (i.e., number of predators required to capture a prey) have captured a prey or after 200 timesteps have passed.

Transition Function. Agents are allowed to move one grid space per timestep. Moreover, predators can either select a capture action or a movement action (not both in the same timestep). Once a sub-team of predators have successfully captured a prey, they are all removed from the game grid for the remainder of the episode. Each agent occupies one grid square, and no agents may pass through

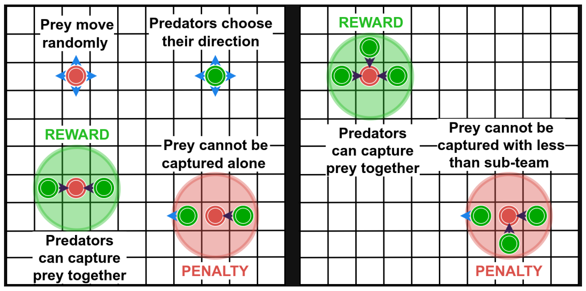
Prey move randomly
Predators choose their direction
REWARD
Predators can capture prey together
Prey cannot be captured alone
PENALTY
REWARD
Predators can capture prey together
Prey cannot be captured with less than sub-team
PENALTY

Figure 1: Synchronized Predator-Prey Task. Blue arrows denote movement (neutral) actions and purple arrows denote capture (synchronization) actions.

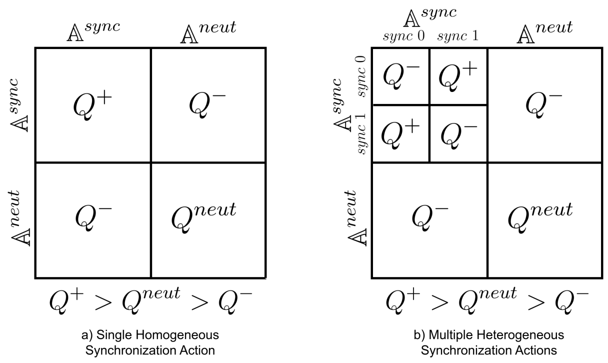

Figure 2: Visual representation of the payoff relationship in the Synchronized Predator-Prey Task for 2-agent sub-teams. Of note are the strict inequalities between the possible Q-values.

each other. The predator agents' actions are controlled by an RL policy. While the prey agents' randomly select their movement from available open positions, or they remain still if no open surrounding position is available. Unattainable actions, such as moving into an occupied grid position, capturing without being adjacent to a prey, or crossing the grid boundary, are considered unavailable.

Actions. All agents have the following five actions: {remain still, left, right, down, up}, which are considered the *neutral actions* ($A^{i,\ neut}$). Additionally, predators have a sixth action, a homogeneous capture action for capturing prey agents. To capture, predators adjacent to a prey must synchronize their capture actions within the same timestep, which is the *synchronization action* ($A^{i,\ sync}$).

Observations. The observation of a predator consists of a 5×5 sub-grid centered around itself, with one channel showing other predators represented by their id and another indicating prey. The observation also includes the position of the predator (itself) on the grid. If a predator has been removed after a successful capture they receive an observation of all zeros for the remainder of an episode.

Reward. A successful capture of a prey is rewarded with a +10, but unsuccessful attempts by less than the required sub-team size are penalized with a -2 miscapture penalty. The maximum reward is dependent upon the total number of possible sub-teams (e.g., 40 for four 2-agent sub-teams and 30 for three 3-agent sub-teams). The reward is shared by all the predators.

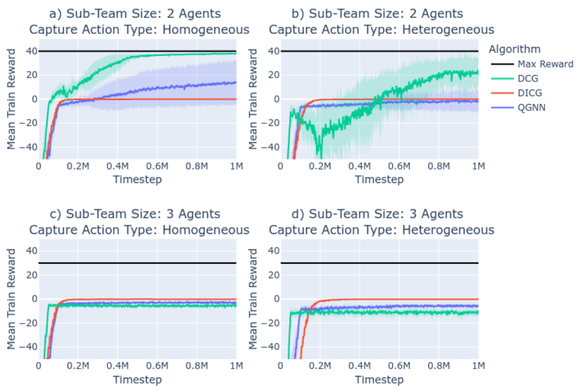

Figure 3: Train episode reward (Mean and shaded Standard Deviation) for ten independently trained models. The *Full* CG topology was used, except for DICG which used an *Attention* mechanism to create the graph. Note that SOTA methods did not solve MSTs well. Currently, DCG is the best solution (i.e., (a) and (b)) but cannot scale to larger sub-teams or handle more complex coordination (i.e., (c) and (d)).

Heterogeneous Synchronized Predator-Prey. A variant of Synchronized Predator-Prey introduces heterogeneous capture actions. For the heterogeneous variant, predators adjacent to the prey must synchronize **unique** capture actions within the same timestep. The number of available capture actions is equal to the number of predators required to capture a prey (i.e., 2 for sub-teams of two and 3 for sub-teams of three), which are the *synchronization actions* ($A^{i,\ sync}$).

We now show that Synchronized Predator-Prey satisfies the MST definition presented in Section 3.2. Consider the system states shown in Figure 1. In each case, $\mathbb{A}^{+}(s)$ is non-empty because there are two sub-teams of agents for which the execution of synchronization actions (as defined above) is optimal, as this would allow for the capture of the prey. Additionally, $\mathbb{A}^{-}(s)$ is non-empty because there are possible scenarios in which one agent might execute a synchronization action while the others do not, and doing so would lead to a large penalty, which is worse than if all the agents had executed neutral actions (as defined above). Therefore, the Synchronized Predator-Prey task meets the requirements of an MST as defined in Section 3.2.

Figure 2 provides a visual representation of the payoff relationship when considering 2-agent sub-teams when they are in a state where successful synchronization is possible. Figure 2a shows the achievable Q-values dependent on the actions the agents select when they are using a single homogeneous capture action. Figure 2b shows the achievable Q-values dependent on the actions the agents select when they are using heterogeneous capture actions.

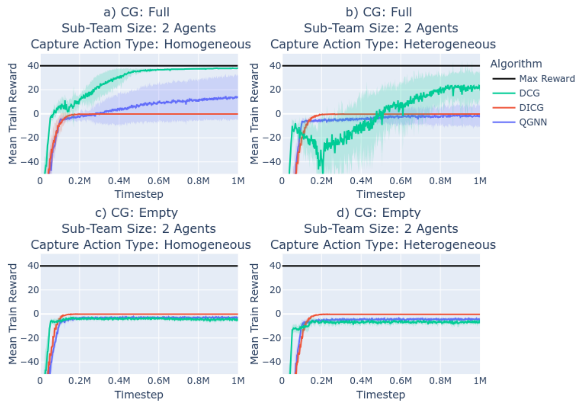

Figure 4: Train episode reward (Mean and shaded Standard Deviation) for ten independently trained models. The *Full* CG topology was used for (a) and (b), except for DICG which used an *Attention* mechanism to create the graph. For (c) and (d), the *Empty* CG topology was used for all algorithms. Note that communication is necessary for MSTs but is not sufficient when complexity increases.

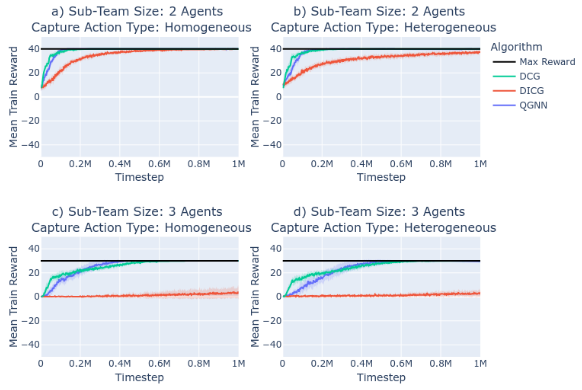

Figure 5: Train episode reward (Mean and shaded Standard Deviation) for ten independently trained models. The *miscapture penalty* was disabled for these training iterations. The *Full* CG topology was used, except for DICG which used an *Attention* mechanism to create the graph. Note with the miscapture penalty disabled the task no longer satisfies the requirements for an MST and was solved by all SOTA methods.

4 EXPERIMENTS

To evaluate the algorithms introduced in Section 2 in MSTs, we conducted experiments using the *Synchronized Predator-Prey* task in four different configurations. The experimental configurations are as follows: (1) 2-agent sub-teams with homogeneous capture action, (2) 2-agent sub-teams with heterogeneous capture actions, (3) 3-predator sub-teams with homogeneous capture action, and (4) 3-predator sub-teams with heterogeneous capture actions. In addition, we used the following Coordination Graph (CG) topologies for our evaluations: *Full* - each agent has an edge connection to every other agent, *Empty* - no edge connections, and *Attention* - a special case for DICG where an attention network generates the coordination graph at each timestep given the state of the environment.

Figure 3 presents the results of our experiments with *Full* CG topology for DCG and QGNN, and *Attention* used for DICG. In general, the results showed that current SOTA methods utilizing coordination graphs had limited success with MSTs (see Figure 3a & 3b). Out of all the approaches evaluated, DCG performed the best with MSTs. DCG showed strong performance with 2-agent sub-teams and homogeneous capture actions but had marginal performance with heterogeneous capture actions (see Figure 3a & 3b). While QGNN showed marginal performance in the 2-agent sub-team with homogeneous capture actions condition (see Figure 3a), it failed in the the other conditions Figure 3b-3d). Note these results differ from those presented in DICG [9] and QGNN [8] as they were not using an MST for their experiments.

Figure 4 represents performance of the SOTA methods on MSTs with and without communication. For the experiments with communication, the *Full* CG topology for DCG and QGNN, and *Attention* used for DICG. In contrast, experiments without communication utilized the *Empty* CG topology. The results provide empirical evidence that some form of communication is needed in order to solve an MST (compare Figure 4a & 4b to 4c & 4d). These results suggest communication is necessary to solve an MST. The communication method used must allow for agents to pass along the information that will allow them to correctly synchronize their actions.

Disabling the miscapture penalty was explored to confirm that the SOTA methods could solve a related non-MST task. It should be noted that the modified task is no longer an MST as defined in Section 3.2, as there is no difference in the Q-values when taking *neutral actions* versus actions that would have been considered *synchronization-negative* in the unmodified (MST) task. Figure 5 shows the results of our experiments using *Full* CG topology for DCG and QGNN, and *Attention* used for DICG. The data suggests that all considered SOTA methods solved the *Synchronized Predator-Prey* task when not constrained by the requirements of an MST (see Figure 5a-5d). However, DICG was unable to perform when the sub-team size was greater than two (see Figure 5c & 5d).

Together the results show the following: (1) DCG performs the best on MSTs, though it is unable to scale to more than 2-agent sub-teams or handle increased coordination complexity (i.e. heterogeneous capture actions), (2) communication is required to solve an MST, and (3) a penalty is critical to the definition of an MST.

5 DISCUSSION AND CONCLUSION

In this paper, we introduced and defined Multi-Agent Synchronization Tasks (MSTs), a novel multi-agent task subset aimed at assessing the capabilities of existing methods in addressing coordination challenges within multi-agent systems. Specifically, we introduced a distinctive Multi-Agent Synchronization Task termed

"Synchronized Predator-Prey," designed to evaluate the effectiveness of prevailing techniques in handling complex coordination scenarios.

Our evaluation of SOTA methods on the Synchronized Predator-Prey task reveals that, currently, no approaches demonstrate robust performance in solving MSTs. Notably, DCG exhibits promising results (see Figure 3a & 3b, albeit failing when coordination complexity increases (i.e. for 3-agent sub-teams and/or heterogeneous synchronization tasks - see Figure 3c & 3d).

Three critical questions and our hypothesized solutions are presented here based on our experimental analysis. First, given the success of DCG in MSTs why does it fail to scale (i.e., to 3-agent sub-teams)? We believe DCG fails to scale mainly because it only models pairwise payoff factors. Thus, DCG cannot fully represent the coordination relationship between groups of more than 2-agents. Second, why does DCG struggle to handle heterogeneous capture actions? We currently hypothesize that DCG struggles with heterogeneous capture actions because the action synchronization facilitated by Max-Plus message passing [7] cannot fully address the need to coordinate unique actions between partners. Third, why do methods (DICG & QGNN) utilizing GNNs for message passing fail to solve Multi-Agent Synchronization Tasks? Our hypothesis posits that a loss of representational capacity occurs with the transition from Max-Plus message passing to the GNN message passing. The lack of representative capacity may result from no longer having the payoff function or a failure of the generic GNN messaging function.

Finally, the results raise questions about the adaptability of GNNs in capturing the intricacies of complex coordination tasks and prompt further exploration into the underlying causes of their limitations in this context. Our ongoing research agenda includes delving into message passing frameworks to find a solution to the representational capacity problem we have identified with our newly defined multi-agent sub-domains we call MSTs. Overall, by asking these questions we aim to deepen our understanding of the challenges posed by MSTs and guide future developments in multi-agent coordination methodologies.

REFERENCES

[1] Derrik Asher, Michael Garber-Barron, Sebastian Rodriguez, Erin Zaroukian, and Nicholas Waytowich. 2019. Multi-agent coordination profiles through state space perturbations. In *2019 international conference on computational science and computational intelligence (CSCI)*. IEEE, 249–252.

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

[3] Jacopo Castellini, Frans A Oliehoek, Rahul Savani, and Shimon Whiteson. 2019. The Representational Capacity of Action-Value Networks for Multi-Agent Reinforcement Learning. In *AAMAS 2019: The 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 1862–1864.

[4] Rolando Fernandez, Erin Zaroukian, James D Humann, Brandon Perelman, Michael R Dorothy, Sebastian S Rodriguez, and Derrik E Asher. 2021. Emergent heterogeneous strategies from homogeneous capabilities in multi-agent systems. In *Advances in Artificial Intelligence and Applied Cognitive Computing: Proceedings from ICAI'20 and ACC'20*. Springer, 491–498.

[5] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. 2002. Coordinated reinforcement learning. In *ICML*, Vol. 2. Citeseer, 227–234.

[6] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[7] Jelle R Kok and Nikos Vlassis. 2006. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *RoboCup 2005: Robot Soccer World Cup IX 9*. Springer, 1–12.

[8] Ryan Kortvelesy and Amanda Prorok. 2022. QGNN: Value Function Factorisation with Graph Neural Networks. *arXiv preprint arXiv:2205.13005* (2022).

[9] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. 2021. Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 764–772.

[10] Ryan Lowe, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[11] Sylvie Morice, Sylvain Pincebourde, Frédéric Darboux, Wilfried Kaiser, and Jérôme Casas. 2013. Predator–prey pursuit–evasion games in structurally complex environments. *Integrative and comparative biology* 53, 5 (2013), 767–779.

[12] Frans A Oliehoek, Christopher Amato, et al. 2016. *A concise introduction to decentralized POMDPs*. Vol. 1. Springer.

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

[14] 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. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

[15] Brian Skyrms. 2001. The stag hunt. In *Proceedings and Addresses of the American Philosophical Association*, Vol. 75. JSTOR, 31–41.