2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Impact of Relational Networks in Multi-Agent Learning: A Value-Based Factorization View

Yasin Findik[1], Paul Robinette[2], Kshitij Jerath[3], and S. Reza Ahmadzadeh[1]

Abstract— **Effective coordination and cooperation among agents are crucial for accomplishing individual or shared objectives in multi-agent systems. In many real-world multi-agent systems, agents possess varying abilities and constraints, making it necessary to prioritize agents based on their specific properties to ensure successful coordination and cooperation within the team. However, most existing cooperative multi-agent algorithms do not take into account these individual differences, and lack an effective mechanism to guide coordination strategies. We propose a novel multi-agent learning approach that incorporates relationship awareness into value-based factorization methods. Given a relational network, our approach utilizes inter-agents relationships to discover new team behaviors by prioritizing certain agents over other, accounting for differences between them in cooperative tasks. We evaluated the effectiveness of our proposed approach by conducting fifteen experiments in two different environments. The results demonstrate that our proposed algorithm can influence and shape team behavior, guide cooperation strategies, and expedite agent learning. Therefore, our approach shows promise for use in multi-agent systems, especially when agents have diverse properties.**

I. Introduction

Multi-agent scenarios are ubiquitous in various domains such as search and rescue operations [1], autonomous driving [2], [3], and multiplayer games [4]. In such scenarios, the success of achieving shared or individual goals critically depends on the coordination and cooperation between agents [5]. Multi-Agent Reinforcement Learning (MARL) approaches have emerged as a popular solution to address the general challenges of cooperation in multi-agent environments. Yet, cooperative MARL faces several challenges that are intrinsic to multi-agent systems, such as the curse of dimensionality [6], [7], non-stationarity [5], and global exploration [8]. Furthermore, the presence of agents having constraints (e.g., limited battery life, restricted mobility) or distinct roles exacerbates these challenges.

Among the various MARL methods for cooperative tasks, the Centralized Training with Decentralized Execution (CTDE) [9] paradigm has become very prominent for addressing numerous cooperative challenges, including but not limited to the ones previously mentioned. While CTDE-based approaches have demonstrated state-of-the-art performance in tackling coordination tasks [10], they lack the ability to steer the coordination strategy that the algorithms converge to. This limitation arises from the fact that these approaches (a) assume agents are identical within the team and (b) do not incorporate any mechanism for specifying the inter-agent relationships. To address this issue, some algorithms have adopted the coordination graph concept [11] to enable the sharing of action-values among the agents using deep learning [12]. However, they require to learn weighing and distributing action-values from a given undirected graph, which may not fully consider the influence of certain behaviors over others. Overall, the problem of agents possessing diverse attributes, a team structure, or a notion of priority among agents remains unresolved.

In this paper, we propose a novel framework that leverages a relational network to capture the relative importance and priorities that agents assign to one another, enabling agents to uncover new cooperation strategies. Our framework uses CTDE paradigm to solve cooperative MARL challenges. We evaluated our approach using fifteen experiments in two distinct environments, each featuring a cooperative task among multiple agents. We have compared our method against the state of the art algorithm, Value Decomposition Networks (VDN) [13]. Our results and comparisons support three main findings. First, unlike other methods, our approach is highly effective in steering the agents towards a specific team behavior, as governed by the relational network. Second, the experiments reveal that our framework enables agents to uncover new behaviors that promote successful task completion, while also preserving the essential structure of the team. Lastly, our results indicate that the proposed approach accelerates the learning of team behavior, particularly in scenarios where agents face constraints, and relational networks play a critical role.

II. Related Work

Multi-Agent Reinforcement Learning (MARL) has gained significant attention in recent years as an active research area, particularly in cooperative settings. Various approaches have been explored to train agents to cooperate effectively towards a shared goal. One such approach is fully centralized learning, where a single controller is shared among all agents, enabling them to learn a joint policy or value function in unison [14]. Despite the potential advantages of fully centralized learning, this approach can be computationally expensive and can lead to intractability issues due to the exponential growth of the observation and action space with the increasing number of agents. Alternatively, fully decentralized learning is another approach to MARL in cooperative settings, where

[1] PeARL Lab, Richard Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA yasin_findik@student.uml.edu, reza@cs.uml.edu
[2] Department of Electrical and Computer Engineering, University of Massachusetts Lowell, MA, USA paul_robinette@uml.edu
[3] Department of Mechanical Engineering, University of Massachusetts Lowell, MA, USA kshitij_jerath@uml.edu

each agent independently learns its own policy. In such approaches, the cooperative behavior arises from the learned policies as they are applied in the environment. For instance, Independent Q-Learning (IQL) [15] trains each agent using a separate action-value table via Q-learning [16]. Since tabular Q-learning is insufficient for addressing high-dimensional state and action space, IQL framework was later extended with function approximation [17]. Yet, independent learning in multi-agent settings is susceptible to non-stationarity, stemming from other agents' actions from the perspective of one agent. Due to the invalidity of the Markov property in non-stationary environments, the convergence of decentralized algorithms based on Q-learning cannot be ensured [18]. Later, the introduction of fully decentralized algorithms featuring networked agents [19] capable of communicating and exchanging information has emerged as a promising approach for addressing the non-stationarity problem. In such frameworks, the connections between agents are governed by an undirected graph. In our approach, we also utilize a (directed) graph to represent the rapport among the agents. However, in contrast to frameworks that prioritize the use of a communication channel [19], our graph serves to symbolize and establish relations between agents, resulting in a relational network that directly influences team behavior.

To overcome the limitations associated with fully centralized and fully decentralized learning in cooperative MARL scenarios, a novel approach has been proposed known as Centralized Training with Decentralized Execution (CTDE) [9]. This paradigm enables individual agents to execute their actions while a centralized mechanism integrates their strategies, thereby ensuring effective coordination and alignment towards a shared goal. The CTDE paradigm has been implemented using both policy-based and value-based methods. Specifically, policy-based methods, such as Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [20] and Multi-Agent Proximal Policy Optimization (MAPPO) [21], incorporate a critic that can take into account the global observations of all agents. On the other hand, value-based techniques, including Value Decomposition Networks (VDN) [13], QMIX [22], and QTRAN [23], enhance Q-Learning by integrating a centralized function that calculates the joint Q-value based on the individual action-values of each agent. These techniques have been shown to be effective in addressing the challenges of multi-agent coordination and achieving higher performance in various scenarios.

Prior research in MARL has predominantly concentrated on achieving an optimal solution for cooperation problems, ranging from fully centralized learning to CTDE paradigm. Yet, the attainment of an optimal or sub-optimal solution by a multi-agent team is intrinsically contingent upon the underlying dynamics of interaction and collaboration among agents [24], [25]. While they utilize reward sharing among agents to adapt individual rewards based on their relationships represented as a relational network, our method involves adjusting the agents' contribution to the team reward. This adjustment is specifically determined by the relational network and does not rely on reward sharing among agents. Ultimately, our method ensures that agents still receive the same individual reward provided by the environment.

Our study proposes a framework that allows for the integration of relational networks into the CTDE approaches to gain a deeper understanding of the impact of inter-agent relationships on cooperation strategies and team interactions. Specifically, our investigation focuses on the context of value-based factorization. However, it is worth noting that the use of relational networks, as proposed in our study, can be extended to other CTDE algorithms.

III. Background

A. Markov Decision Process

We characterized Decentralized Markov Decision Process as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$ where $s \in \mathcal{S}$ indicates the true state of the environment, the joint set of individual actions and rewards are represented by $\mathcal{A} \coloneqq \{a_1, a_2, \ldots, a_n\}$, $\mathcal{R} \coloneqq \{r_1, r_2, \ldots, r_n\}$, respectively, $\mathcal{T}(s, A, s')\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [1, 0]$ is the dynamics function defining the transition probability, and $\gamma \in [0, 1)$ is the discount factor.

B. Deep Q-Learning

The Deep Q-Network (DQN) [26] is a variant of an action-value method, which defines $Q(s, a) = \mathbb{E}[G|S = s, A = a]$ where G denotes the return, that employs a deep neural network to approximate the Q-function, represented as $\hat{Q}(s, a, \theta)$ where θ is the weight vector. Unlike tabular Q-learning, DQN is capable of effectively handling high-dimensional state and action spaces by leveraging deep learning. However, the training of DQN poses a significant challenge due to instability and divergence caused by the Q-network parameters being updated in each step, which violates the assumption of independently and identically distributed (i.i.d) data points. To address these challenges, Mnih et al. [26] have also proposed several techniques, including experience replay and fixed Q-target networks, which have since become standard in many deep reinforcement learning algorithms.

In a nutshell, two deep Q-neural networks are used for each Q-function, namely: Prediction Neural Network (P-NN), and fixed Target Neural Network (T-NN) which is the P-NN from a previous iteration. Also, a replay memory is utilized to keep a large number of transitions experienced by the agent during its interactions with the environment. Each transition consists of a tuple $\langle s, a, r, s' \rangle$. To train the P-NN, a batch of transitions of size b is sampled from memory. The Temporal Difference (TD) error is calculated between the P-NN value and T-NN value, as follows:

$$e_{\text{TD}} = \sum_{i=1}^{b} [r + \gamma \max_{a'}(Q(s', a', \theta_t)) - Q(s, a, \theta_p)], \quad (1)$$

where θ_p are the weights of the P-NN, and θ_t are those of the T-NN which is regularly updated with θ_p. The parameters of the P-NN are updated using an optimizer in the direction of reducing e_{TD}. In centralized multi-agent systems, a single Q-function is employed for the meta-agent, providing access to

the observations of all agents, whereas, in fully decentralized systems, each agent learns its Q-function independently using DQN.

C. Value Function Factorization

Value function factorization methods, which adhere to the CTDE paradigm, have been proposed as an efficient approach for handling a joint action-value function, whose complexity increases exponentially with the number of agents. These methods effectively address the non-stationarity problem of decentralized learning through centralized training and overcome the scalability issue of centralized learning through decentralized execution. VDN [13] and QMIX [22] are two exemplary methods for factorizing value functions.

VDN and QMIX both maintain a separate action-value function Q for each agent $i \in \{1, ..., n\}$. They merge these individual Q_i values to obtain the central action value Q_{tot} using additivity and monotonicity, respectively. Specifically, VDN sums Q_is to obtain Q_{tot}, as

$$Q_{\text{tot}} = \sum_{i=1}^{n} Q_i(s, a_i),$$

while QMIX combines them using a state-dependent continuous monotonic function, as follows:

$$Q_{\text{tot}} = f_s(Q_1(s, a_1), ..., Q_n(s, a_n)),$$

where $\frac{\partial f_s}{\partial Q_i} \geq 0, \forall i \in \{1, ..., n\}$.

These value function factorization methods most commonly utilize DQN to approximate the action value function. Different from decentralized learning where all agents' individual P-NNs are trained independently by using (1), or centralized learning where a single P-NN exists and is trained with (1) by providing access to the observations of all agents, in these value function factorization methods, Q_{tot} is used to compute e_{TD} by updating (1) as follows:

$$e_{\text{TD}} = \sum_{i=1}^{b} [r_{\text{team}} + \gamma \max_{u'}(Q_{\text{tot}}(s', u', \theta_t)) - Q_{\text{tot}}(s, u, \theta_p)], \tag{2}$$

where r_{team} is the sum of agents' rewards with uniform weights, u is the joint action of the agents. The agents' P-NN are trained using an optimizer in the direction reducing obtained e_{TD}. The coordination of agent actions towards maximizing the team reward is facilitated by this process. Consequently, the central aspect of the CTDE paradigm becomes apparent: the agent networks are trained using a centralized Q_{tot}, while the actions of each agent are determined by its own neural network, rendering the execution decentralized.

IV. Proposed Method

In cooperative MARL, the presence different team structures often yield multiple solutions with varying optimality. Value factorization methods and others, strive to maximize team rewards and converge towards one of several solutions,

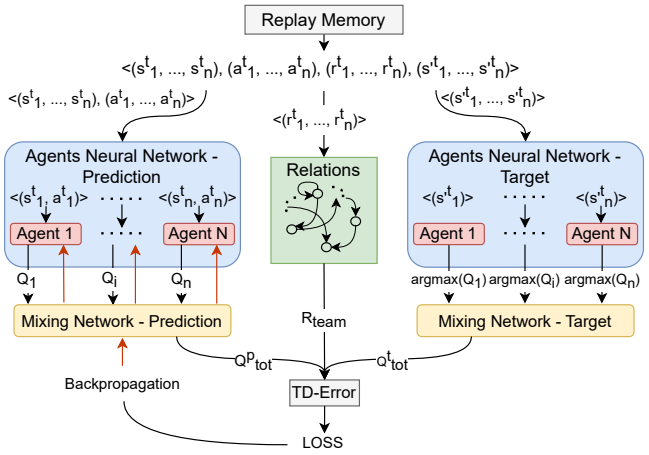


Fig. 1: The overall relationship awareness framework. In blue are the agents' P-NNs and T-NNs. The relational network is represented in green. Mixing networks are used combine the individual action-values to obtain the central action-value.

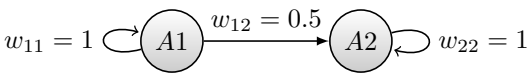
possibly achieving the global optimum. However, stochasticity in agents' exploration can influence convergence towards a particular team behavior if multiple cooperation strategies exist with the same maximum total reward. Furthermore, in real-world settings, individual agents/robots may be subject to varying constraints or possess distinctive attributes that significantly influence overall team behavior, making it challenging to learn a viable solution without a deeper understanding of the team structure. We propose a novel framework, depicted in Fig. 1, that incorporates relationship awareness into the agents' learning algorithm. And, we opt to explore and study the framework using VDN, as it is a simple yet effective CTDE approach for learning cooperative behaviors, leading us to introduce Relationship-Aware VDN (RA-VDN). RA-VDN is a generalization over VDN that enhances team behavior, provides control over optimal solutions, and expedites agent learning by utilizing the relationships between agents.

The inter-agent relationships are represented by a relational network which is utilized to integrate the individual rewards to obtain the team reward used in (2). The MDP, defined in III-A, is modified to include the relational network which is characterized by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ where each agent $i \in \{1, ..., n\}$ is a vertex v_i, $\mathcal{E}$ is the set of directed edges e_{ij} directed from v_i to v_j, and the weight of the edges is represented by the matrix $\mathcal{W}$ with elements $w_{ij} \in [0, 1]$ for each edge. The direction and weight of each edge signify the relationship between the agents. An edge e_{ij} directed from i to j indicates that agent i cares about or is vested in the outcomes for agent j. Based on the modified MDP represented as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{G}, \gamma \rangle$, the team reward defined as follows:

$$r_{\text{team}} = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{E}_i} w_{ij} r_j,$$

where $\mathcal{E}_i$ denotes the set of vertex indices that have an edge directed from v_i, and r_j is the reward of the agent represented

by v_j. For example, the directed graph as follows:



$$R_{\text{team}} = r_1 + 1.5r_2$$

illustrates the relationships between two agents. Examining the team reward obtained using this relational network reveals that the second agent's contribution to the team reward is increased by adding half of the reward second agent receives with the assistance of first agent. By giving more emphasis on the second agents, the algorithm can produce various emergent behaviors, such as the first agent helping the second agent in reaching its goal, or sacrificing own individual reward entirely to increase the second agent's reward. Furthermore, it can accelerates the learning of team behavior by agents.

Similar to other CTDE approaches, in VDN, the primary objective of each agent i is to obtain a policy, that optimizes the team performance. The learned policy is then employed to determine the optimal action of the agent a_i for a given state s, as follows:

$$a_i = \pi_i^{\text{VDN}}(s) = \max_a(Q_i(s, a, \theta_p)),$$

where π_i represents the policy of agent i, and Q_i is represented using DQN to approximate its action-value function. For all agents, the notation takes the following form:

$$u = \pi^{\text{VDN}}(s) = \max_u(Q(s, u, \theta_p)),$$

where u is a vector, containing the actions of all the agents and π is a vector of policies, where each element π_i represents the policy of the respective agent i. We formulate RA-VDN, as follows:

$$u = \pi^{\text{RA-VDN}}(s|\mathcal{G}) = \max_u(Q(s, u, \theta_p|\mathcal{G})),$$

where the policy is conditioned by the relational network $\mathcal{G}$.

We would like to highlight that in cases where $\mathcal{G}$ represents a self-interest relational network the policies for RA-VDN and VDN become equivalent. That is because in VDN, each agent contributes their reward equally to the team reward.

$$\pi^{\text{RA-VDN}}(s|\mathcal{G}) \equiv \pi^{\text{VDN}}(s),$$

where $\mathcal{G} \in \{\text{a}, \text{d}, \text{h}\}$ networks depicted in Fig. 3. Overall, RA-VDN allows for the shaping and influencing of team behavior among agents utilizing different relational networks, whereas VDN leads to agents randomly converging to one of multiple solutions without any control over the cooperation strategies. Also, noted from the experiments, it accelerates convergence of the agents to the team behavior where agents have constraints.

V. Experiments

A. Environments

To assess the efficacy of the proposed approach in shaping and enhancing team behaviors, we applied the RA-VDN algorithm to two environments: a multi-agent grid-world environment, and the *Switch* environment proposed in [27].

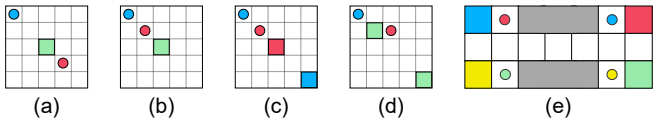


Fig. 2: (a-d) multi-agent grid-world environment. (e) *Switch* environment with four agents. In the three-agent setup, the yellow agent and its goal location were removed. For the two-agent setup, the green agent and its goal location were in addition removed.

1) Multi-agent Grid-world Environment: We created a 5x5 grid-world environment with resources and two agents, colored red and blue (see Fig. 2(a-d)). The resources are categorized based on their color, green resources are consumable by both agents, while red and blue resources can only be consumed by their respective colored agents. The objective of each episode is for the agents to consume all the resources by visiting their locations, with each agent limited to one resource per episode. The agents have five possible actions: move up, down, left, right, or stay still. They can also *push* each other if they are adjacent and the pushing agent takes a non-idle action towards the pushed agent, which must be idle. After a *push*, the pushing agent remains in place while the other agent moves one space in the pushed direction. Consuming a resource yields $+10$ reward, while the agents are penalized with -1 for each time step per unconsumed resource unless they occupy a resource location which serves as a safe spot. The episode ends when all resources are consumed or the maximum time steps are reached. We designed this environment to challenge agents with real-world constraints, such as limited battery life and restricted mobility, and to demonstrate how relational networks can help achieve cooperative tasks by prioritizing based on the agents' constraints.

2) Switch Environment: Figure 2(e) illustrates the *Switch* environment, which comprises a grid-world divided into two regions connected by a narrow bridge. The environment accommodates up to four agents with distinct colors, each aiming to reach their respective goal boxes of matching color on the opposite end. This is challenging due to the narrow bridge, which allows only one agent to cross at a time. Agents can move left, right, up, down, or stay still, and receive a reward of $+5$ upon reaching their goal, but incur a penalty of -0.1 per time step away from it. The episode ends after 50 time steps or when all agents reach their goals, requiring collaborative efforts among agents to maximize rewards and prevent obstructing each other's paths. We select this environment as it offers various cooperation strategies and facilitates evaluating relational networks. Additionally, it was employed in the original VDN paper, enabling a lucid comparison between our approach and VDN.

B. Models and Hyperparameters

In setting our algorithm up for both environments, we employed a Multi-Layer Perceptron (MLP) with two hidden layers, each consisting of 128 neurons and utilizing the ReLU activation function. The prediction model for each agent is

trained 10 times per episode using uniformly drawn batches of size $b = 32$ selected from the replay memory, which has a capacity of 50k time-steps. We used *Adam* optimizer with a learning rate of 0.001 and the loss function is the squared TD-error. The weights of the target network are updated with those of the prediction network every 200 episodes. To promote exploration, an ε-greedy method is applied, where ε is linearly annealed over time. Finally, the discount factor, γ was set to 0.99.

The results of the experiments for both environments, presented in Fig. 4 and 5, illustrate the average training reward over 10 runs as represented by the shaded regions, and the average test rewards of the agents, denoted by the solid lines. The test rewards were determined by evaluating the individual rewards of agents based on a greedy strategy, with the training process being interrupted every 50 episodes.

C. Results for Multi-agent Grid-world Environment

We evaluated the performance of RA-VDN through eight independent experiments across four distinct configurations utilizing various relational networks within this environment, and compare the individual rewards obtained by the agents.

Resource Collection (RC) Scenario: This scenario comprises of two agents, represented by red and blue circles and an undedicated resource depicted by a green rectangle, as shown in Fig. 2(a). The resource has intentionally been positioned in closer proximity to the red agent. The determination of which agent is going to consume the given resource is contingent upon a relative importance of the agents within the team. When the importance of agents is equal (like in VDN), as in Fig. 3(a), or when the importance of the red agent surpasses that of the blue agent, as in Fig. 3(c), the red agent consumes the resource since resource is located closer to it (see Fig. 4(b)). Nevertheless, by introducing a relational network that assigns greater importance to the blue agent (Fig. 3(b)), the red agent reserves the resource for the blue agent's consumption, as depicted in Fig. 4(a). By enabling the value decomposition algorithm to access the relationships among agents, it becomes feasible for agents to discover novel behaviors.

Resource Collection with Restricted Mobility (RC-RM) Scenario: As illustrated in Fig. 2(b), this experiment involves two agents, with the red agent restricted to vertical movement, and one resource positioned out of reach for the red agent without the help of blue. After training for 25000 episodes using the relationship in Fig. 3(c), where the red agent's significance surpasses the blue agent's, the blue agent assisted the red agent in reaching the resource, resulting in a higher individual reward for the red agent (7.0 ± 0.0) than the blue agent (-3.0 ± 0.3), as shown in Fig. 4(c) and Table I. However, when the relational network in Fig. 3(b) is employed, the red agent deviates from the blue agent's path to the resource, resulting in the blue agent (7.0 ± 0.0) receiving a higher reward than the red agent (-4.0 ± 0.0), as shown in Fig. 4(d). These findings show the significance of the relative importance of agents in determining their behavior and outcomes in team settings.

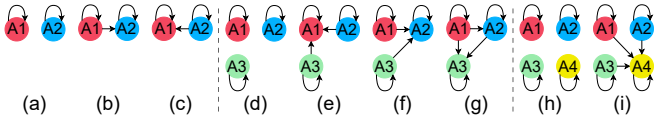


Fig. 3: Relational networks employed in RA-VDN.

Dedicated Resource Collection with Restricted Mobility (DRC-RM) Scenario: In this setup, the environment features two agents and two resources, with each resource dedicated to an agent of the same color, as shown in Fig. 2(c). Similar to the prior experiment, the red agent can only move vertically. The optimal solution entails the blue agent helping the red agent before consuming its assigned resource and only then proceeding to consume its own resource. The findings of training with VDN for 40000 episodes show that the blue agent only learns the optimal solution in 2 out of 10 runs (20%), resulting in the blue agent's reward being (-11.7 ± 11.6) and the red agent's reward being (-7.4 ± 11.1), as depicted in Table I. Yet, when using RA-VDN, the relational network in Fig. 3(c) accelerates the agent learning process, leading to significantly higher rewards for both the red agent (2.4 ± 6.3) and blue agent (0.9 ± 3.1), as shown in Fig. 4(f) and Table I. RA-VDN outperformed VDN by achieving the optimal solution in 80% of runs and yielding higher agent rewards. Overall, this experiment demonstrates how proper relationships can expedite the learning of team behavior when any of the agents need special assistance, and RA-VDN benefits from it.

Resource Collection with Battery Constraint (RC-BC) Scenario: In this experiment, two agents with battery constraints and two undedicated resources are used, as shown in Fig. 2(d). The red and blue agents have a maximum limit of 10 and 5 non-idle actions per episode, respectively, to mimic battery constraints. One resource is located beyond the blue agent's reach due to its limited battery, making the optimal policy for the team to have the red agent to consume the distant resource and leave the nearest one for the blue agent. However, after training VDN for over 10000 episodes, the red agent failed to learn this strategy, as indicated in Fig. 4(g). It learned to relinquish the nearest resource for the blue agent to use as a safe spot after consumption. Unfortunately, this led to the other resource remaining unconsumed since an agent could consume only

TABLE I: Average reward with 95% confidence intervals for ten runs on multi-agent grid environment after training completed.

	RC Scenario		RC-RM Scenario		DRC-RM Scenario		RC-BC Scenario	
Relational Network	Fig. 3(b)	Fig. 3(c)	Fig. 3(c)	Fig. 3(b)	Fig. 3(a)	Fig. 3(c)	Fig. 3(a)	Fig. 3(b)
Red Agent	-4.2 ± 0.4	9.0 ± 0.0	7.0 ± 0.0	-4.0 ± 0.0	-7.4 ± 11.1	2.4 ± 6.3	2.0 ± 1.0	5.0 ± 0.0
Blue Agent	7.0 ± 0.0	-2.0 ± 0.0	-3.0 ± 0.3	7.0 ± 0.0	-11.7 ± 11.6	0.9 ± 3.1	0.3 ± 2.3	7.9 ± 0.2

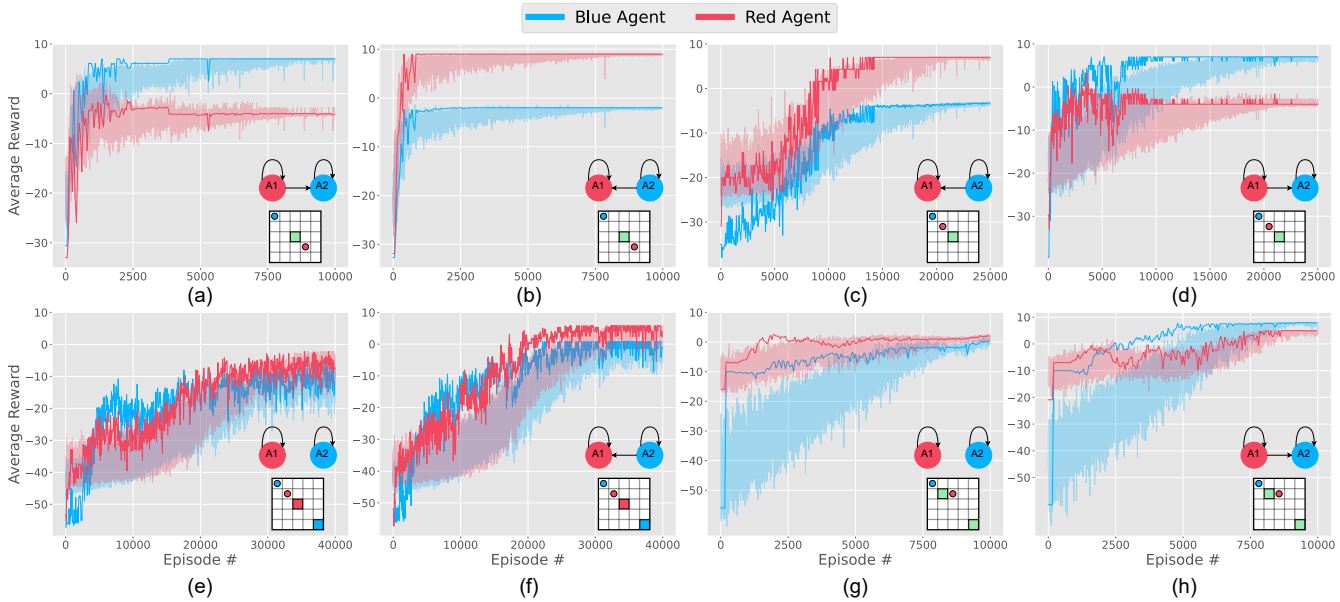


Fig. 4: Multi-agent grid-world environment results with different relational networks (details in Section V-C).

one resource, causing episode termination solely due to the time-step threshold. Yet, by employing RA-VDN with the relational network illustrated in Fig. 3(b), the red agent learns to reserve the nearest resource for the blue agent's consumption and proceed towards the other resource as it cares about the blue agent. As shown in Fig. 4(h) and Table I, the blue agent receives an individual reward of (7.9 ± 0.2), while the red agent gets (5.0 ± 0.0). These results indicate that RA-VDN is capable of finding and converging to the optimal policy, while VDN fails to resolve the environment where agents have different battery life.

D. Results for Switch Environment

We conducted seven independent experiments in the *Switch* environment using two, three, or four agents. Our algorithm was compared against VDN based on both individual and collected rewards by the team. In Fig. 5, collective reward shows the sum of individual rewards attained by the agents (i.e., the team reward maximized in VDN). Note that collective reward differs from team reward used to train RA-VDN, which is computed using a relational network.

Two-Agent Scenario: As depicted in Fig. 2(e), this scenario includes a red and a blue agent with their corresponding goal locations that marked with the same color. Given that only one agent can cross the bridge at a time, there are two optimal policies for this scenario: either the red agent crosses the bridge ahead of the blue agent or the blue agent does. VDN, in each run, converges to one of the two optimal policies because of the algorithm's equal weighting of both agents (i.e., this is the equivalent of the network in Fig. 3(a)), which leads to a randomized selection of optimal policies. As shown in Fig. 5(a) and Table II, the individual rewards for both agents are nearly identical as a result of averaging their rewards over 10 runs. In other words, in each run the agent that crosses the bridge maximizes its individual reward and the average value for both agents become similar as the number of runs increases. For RA-VDN, we introduce two relational networks: first the red agent places importance to the blue agent (Fig. 3(b)); second, the blue agent assigns importance to the red agent (Fig. 3(c)). These networks allow us to investigate team behavior and solution preference. The results in Fig. 5(b) and Table II indicate using RA-VDN with network in Fig. 3(b), the blue agent accumulates higher individual rewards (4.40 ± 0.00) as it is the first to use the bridge and than the red agent (3.80 ± 0.00). Conversely, when the network in Fig. 3(c) was applied, the situation is reversed with the red agent having a higher total reward (4.39 ± 0.02), as depicted in Fig. 5(c). Comparing results from these three experiments reveals that the collective rewards are the same while the relational network influences which *optimal* solution the algorithm converges to.

Three-Agent Scenario: This setup involves three agents positioned as illustrated in Fig. 2(e). The optimal policy, which minimizes waiting time, is for the pair of agents on the left side to cross the bridge initially, as they are heading in the same direction, either in the order of the red–green–blue or green–red–blue. As illustrated in Fig. 5(d) and Table II, VDN experiments show that red and green agents have similar individual rewards $(4.38 \pm 0.04$ and $4.35 \pm 0.02)$ since they randomly alternate taking the lead in passing the bridge over 10 runs, while the blue agent has lower individual reward (3.66 ± 0.04). Yet, RA-VDN has the ability to influence the order through the relational network shown in Fig. 3(e), where the red agent holds the highest significance. Fig. 5(e) and Table II indicate the bridge usage order becomes consistent as red–green–blue, with individual rewards of 4.40 ± 0.00, 4.30 ± 0.00, and 3.70 ± 0.00, respectively.

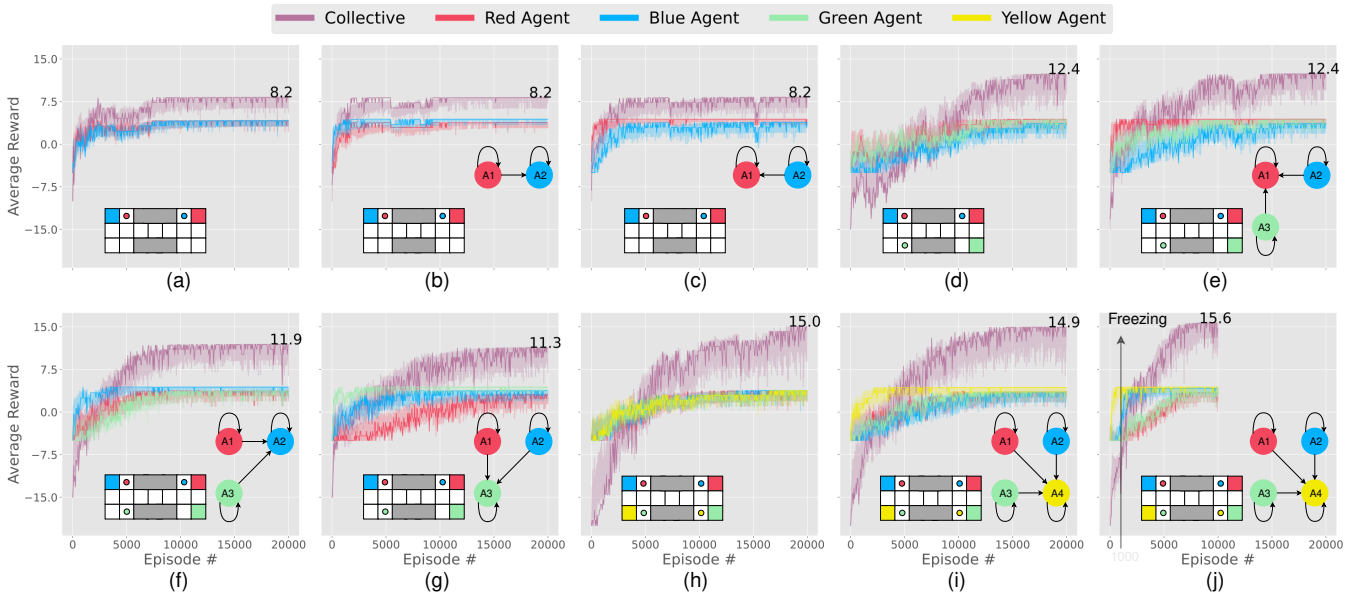


Fig. 5: *Switch* environment results with varying number of agents and different relational networks (details in Section V-D).

Utilizing relational networks can also facilitate the discovery of new behaviors for agents to converge upon, while guiding the algorithm to converge towards an optimal solution. For instance, Fig. 3(f) shows that both the red and green agents prioritize the blue agent, granting it the first turn. Subsequently, the second turn is randomly assigned to either the red or green agent, as their importance weights are equal and they share the same side. The findings of RA-VDN with this network are indicated in Fig. 5(b) and Table II, which reveal that the blue agent has a higher individual reward (4.40 ± 0.00) than the others while the red (3.75 ± 0.03) and green (3.76 ± 0.07) agents have almost the same rewards due to alternating turns. Furthermore, to assess the agents' adherence to relational networks and cooperative strategies over individual and collective rewards, we introduce a novel relational network illustrated in Fig. 3(g). The network elicits the sequential crossing of the bridge as follows: the green agent crosses from the left side, followed by the blue agent from the right, and the red agent crosses from the left, resulting in the red agent waiting for the alternating turn. Fig. 5(g) and Table II confirm that the agents prioritize relationships over collective reward, even at a cost. However, the collective reward is lower here as the red agent sacrifices more by taking the last turn, due to the environment's optimal policy being influenced by the relational network, leading to the convergence of the agents towards the *desired* behavior.

Four-Agent Scenario: In this experimental, four agents are used, with an equal number of agents on each side (Fig. 2(e)). VDN yields a solution that involves stochastic selection of the side that crosses the bridge initially, as well as which agent from that side takes the first turn. The results, depicted in Fig. 5(h) and Table II, demonstrate that the individual rewards derived from VDN are very similar, owing to the stochasticity induced by the existence of multiple optimal policies that equally weight all the agents. Yet, RA-VDN with the relation network illustrated in Fig. 3(i) has a significant effect on the sequence of bridge usage. The yellow agent is given importance by all agents, leading to the right side taking the initial turn and allowing the yellow agent to be the first to cross the bridge, as shown in Fig. 5(i) and Table II. We anticipate that the blue agent passes the bridge second to mitigate the undesirable impact of remaining in the environment, as in three-agent scenario. However, due to the increased complexity of the team network and small amount of negative reward, in 80% of the runs, the blue agent passes the bridge subsequent to the yellow agent, while in the remaining 20%, the bridge is crossed by either of the three agents randomly. The observed behavior leads to the blue, red, and green agents receiving individual rewards that are highly comparable, while the yellow agent attains the highest reward (4.34 ± 0.09).

To address the challenge of the team failing to fully learn the expected team behavior, we propose an extension to

TABLE II: Average reward with 95% confidence intervals for ten runs on *Switch* environment after training completed.

	Two-Agent Scenario			Three-Agent Scenario				Four-Agent Scenario		
	VDN	RA-VDN		VDN	RA-VDN			VDN	RA-VDN	Frozen RA-VDN
Relational Network	N/A	Fig. 3(b)	Fig. 3(c)	N/A	Fig. 3(e)	Fig. 3(f)	Fig. 3(g)	N/A	Fig. 3(i)	Fig. 3(i)
Red Agent	4.10±0.19	3.80±0.00	4.39±0.02	4.38±0.04	4.40±0.00	3.75±0.03	3.13±0.03	3.70±0.34	3.43±0.29	3.47±0.15
Blue Agent	4.15±0.15	4.40±0.00	3.86±0.06	3.66±0.04	3.70±0.00	4.40±0.00	3.78±0.02	3.78±0.25	3.61±0.17	4.20±0.00
Green Agent	—	—	—	4.35±0.02	4.30±0.00	3.76±0.07	4.40±0.00	3.77±0.36	3.54±0.23	3.54±0.14
Yellow Agent	—	—	—	—	—	—	—	3.80±0.30	4.34±0.09	4.40±0.00

our algorithm, which includes freezing the neural network. Initially, the neural network of agents is trained for 1000 episodes, where the primary agent learns to reach its goal, and the others learn to prioritize it. After the initial phase, we freeze the prioritized agent's network (i.e., its parameters no longer become updated) and it becomes a dynamic element of the environment for subsequent 9000 episodes of training, with other agents learning their policies accordingly. Through this approach, the complexity of the problem is reduced, and the learning process is accelerated. The results, presented in Fig. 5(j) and Table II, demonstrate that the yellow agent achieves the highest reward (4.40 ± 0.00) among the four agents because it is the initial agent to utilize the bridge. The blue agent's reward (4.20 ± 0.00) surpasses that of the remaining agents because it can pass the bridge right after the yellow agent, while the individual rewards of the green (3.54 ± 0.14) and red (3.47 ± 0.15) agents are close due to the alternate turns they take.

VI. Conclusion and Future Work

We propose a novel framework that incorporates relationship awareness into agents' learning algorithms. Our framework enables the discovery of new cooperative behaviors and the ability to guide agents' behavior based on inter-agent relationships. Our experiments, conducted in a multi-agent grid-world environment, validate the effectiveness of our approach in solving environments with agents having different abilities. Moreover, in the *Switch* environment, we demonstrated the algorithm's power in influencing agents' behavior within a team. However, we acknowledge that adjusting relational weights can become challenging as the relational network becomes denser with an increasing number of agents. To overcome this issue, we introduce the idea of freezing the agents neural networks, which we show to be effective in reducing problem complexity and expediting the learning process. As a next step, we aim to conduct additional experiments in more complex environments that involve an increased number of agents and compare the performance of our algorithm with other state-of-the-art methods.

VII. Acknowledgments

This work is supported in part by NSF (IIS-2112633) and the Army Research Lab (W911NF20-2-0089).

References

[1] A. Kleiner, J. Prediger, and B. Nebel, "Rfid technology-based exploration and slam for search and rescue," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, p. 4054.

[2] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang Jr, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, 2017.

[3] T. Kim and K. Jerath, "Congestion-aware cooperative adaptive cruise control for mitigation of self-organized traffic jams," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6621–6632, 2022.

[4] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069*, 2017.

[5] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[6] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?" *Artificial intelligence*, vol. 171, no. 7, pp. 365–377, 2007.

[7] H. Haeri, K. Jerath, and J. Leachman, "Thermodynamics-Inspired Macroscopic States of Bounded Swarms," *ASME Letters in Dynamic Systems and Control*, vol. 1, no. 1, 03 2020, 011015. [Online]. Available: https://doi.org/10.1115/1.4046580

[8] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[9] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

[10] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022.

[11] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," *Advances in neural information processing systems*, vol. 14, 2001.

[12] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *International Conference on Machine Learning*. PMLR, 2020, pp. 980–991.

[13] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[14] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[15] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[16] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[17] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, 2017.

[18] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. De Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *arXiv preprint arXiv:1707.09183*, 2017.

[19] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5872–5881.

[20] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[21] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.

[22] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.

[23] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 5887–5896.

[24] B. Baker, "Emergent reciprocity and team formation from randomized uncertain social preferences," *Advances in neural information processing systems*, vol. 33, pp. 15 786–15 799, 2020.

[25] H. Haeri, R. Ahmadzadeh, and K. Jerath, "Reward-sharing relational networks in multi-agent reinforcement learning as a framework for emergent behavior," *arXiv preprint arXiv:2207.05886*, 2022.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] A. Koul, "ma-gym: Collection of multi-agent environments based on openai gym." https://github.com/koulanurag/ma-gym, 2019.