Privacy-Aware Deep RL for Sequential Coalition Formation Decisions under Uncertainty

Gerasimos Koresis, Stergios Plataniotis, Leonidas Bakopoulos, Charilaos Akasiadis, Georgios Chalkiadakis
School of Electrical and Computer Engineering
Technical University of Crete, Chania, Greece

***Abstract*—[1] Task execution in multiagent settings often calls for the formation of coalitions, since agents may possess complementary skills and/or resources that have to be pulled together for a task. Naturally, the formation process and its efficiency is affected by strategic agent choices, as well as by the uncertainty inherent in practically all real-world settings; thus, in principle it can be facilitated by reinforcement learning (RL). The practicality of employing RL for each individual to learn aspects of the coalition formation problem in large open settings is questionable, however, due to the scarcity of interactions among specific agents and also due to privacy concerns. As such, we propose a novel framework that effectively intertwines sequential decision making during coalition formation under uncertainty with deep RL (DRL) techniques designed to fit the needs of such challenging environments and overcome the aforementioned issues. This is facilitated by the assumption of capability-related agent types; and by allowing the joint RL training of agents in ways that do not jeopardize their privacy. We put forward two DRL algorithms that guarantee privacy preservation by design, as is also verified by a theoretical proof that we provide. Our experiments demonstrate the effectiveness of our approach, and its potential for transfer learning, as already trained models can be successfully employed in environments with different reward functions. To the best of our knowledge, ours is the first approach to allow autonomous agents to employ deep RL for sequentially optimal decisions in large open coalitional task allocation settings, while addressing uncertainty and privacy concerns.**

***Index Terms*—Multiagent systems, Machine Learning, Group Privacy**

I. Introduction

Coalition formation is a microeconomics paradigm widely used to facilitate the coordination of autonomous agents and the utilization of their resources to complete joint tasks [1], [2]. In general, coalition formation faces challenges due to *(i)* scalability, when the number of possible coalitions grows rapidly; *(ii)* increased models' complexity in evolving and dynamic environments [3], and *(iii)* communication constraints [4]. Moreover, in dynamic multiagent settings, the efficiency of the formed coalitions may be hindered by inherent uncertainty regarding aspects of the agents or the environment, e.g., agent skills or types, agent contribution capabilities, differences between the expected and realized utility, and so on [3], [5]. Such concepts are relevant to the majority of real-world coalition formation application domains—such as the Smart Grid [6], transportation [7], and robotics [8], among others [9].

[1]Long version of the paper appearing in the Proceedings of the 37th International Conference on Tools with Artificial Intelligence (ICTAI-2025).

Whenever attempting to form coalitions under various forms of uncertainty, it is imperative that agents acquire the means for *learning* the values of their coalitions, and/or the capabilities of their peers [2], [10], [11]. In large open coalitional task allocation environments, which is our domain of interest and in which large numbers of agents can enter or leave the multiagent system at any time and need to form coalitions in order to complete joint tasks, individual learning arguably becomes impractical due to *(a)* the fact that specific agents may be rarely encountered in a given agent's interactions with the system; and, *(b)* the fact that, in general, reduced learning opportunities come up due to potentially long periods of inactivity. Both issues result to rare training instances, thus the learned models do not fit adequately.

In recent years *deep reinforcement learning (DRL)* has emerged as a powerful framework for training agents to take sequential decisions in complex environments. Centralized solutions in DRL [12] involve a central agent or controller that makes decisions for all entities based on a global perspective of the environment. Such centralized DRL approaches have been used for various coalition formation settings [13]–[16], but pose scalability challenges due to communication overheads.

Moreover, the use of shared neural networks insinuates that experiences of agents that are strangers to each other are shared, in order to take part in the training process. This, introduces important privacy concerns—since, e.g. a shared network infrastructure could be exploited for harvesting experiences of other agents [17]. In our domain of interest, in particular, it is arguably unacceptable to assume that autonomous agents taking strategic sequential decisions in task allocation settings could learn via experiences accumulated by agents essentially competing with them for reward accumulation (albeit for reward accrued by coalitional task execution). At the same time, however, it is probable that in real-life task allocation settings multiple autonomous agents either belong to a single owner entity (e.g. a parent organization or company that participates via its subsidiaries in calls-for-proposals domains [2], [18]); and, moreover, agents might be able to share information via some form of a social network without violating privacy. This opens up the possibility of allowing agents within the same *"social group"* to utilize group-dedicated neural networks, or just some shared layers of such, to enhance their learning process and increase efficiency.

Against this background, we propose a *deep reinforcement learning (DRL)* methodology for learning the uncertain value

of coalitions in large task allocation environments, where agents have to take sequential decisions regarding distributing parts of their resources among different coalitions. Interestingly, inspired by other works utilizing shared layers of neural networks [19], we modify DQN [20] in a manner where each *"social group"* uses a specific subset of the neural network's layers, giving rise to a DRL technique we term as *Privacy-Aware Multihead-DQN (PAMH-DQN)*. This preserves the privacy among agents of distinct social groups, since we ensure that each head is *dedicated* to its *"social group"*—i.e., it has full access to experiences attained by members of its associated *"social group"* alone, while also training on shared network weights that effectively anonymize and summarize other groups' experiences. Now, this privacy-preserving sharing of anonymized experiences, on the one hand has the potential to increase learning efficiency. On the other hand, basing one's training even partially on data produced by others' interactions, could affect performance negatively (due to possibly "confusing" the neural network learning). That is why we also implement a *Multiple-DQN* algorithm that approaches privacy concerns differently, employing a separate DQN for each social group, ensuring that the learning of a social group is not influenced in any way by the experiences and learning of the other groups.

Our contributions are as follows. First, we propose a novel framework for the learning of coalition values within large open environments characterized by diverse agent types. Importantly, our framework intertwines DRL learning of coalitional values, with *any* coalition formation protocol of choice that requires sequential formation decisions. This enables DRL agents to take sequentially optimal formation decisions under uncertainty, in distinction to previous work in the field. Second, our framework introduces a social network perspective to coalition value learning, respecting privacy concerns regarding the information shared among agents in such environments. Third, to this end, we put forward two privacy-respecting methods; one of them, *PAMH-DQN*, builds on Bootstrapped-DQN [19] to make a novel use of the network heads. Fourth, we provide a theoretical proof for the fact that *PAMH-DQN* preserves privacy among social groups. Fifth, our experimental evaluation verifies the effectiveness of our approach for *sequential* coalitional task allocation in large open environments under uncertainty; while demonstrating its potential for allowing already trained models to be used in environments with different coalitional reward functions.

II. Background and Related Work

Building on DQN [20], *Bootstrapped DQN* [19] incorporates the concept of ensemble learning, thus enhancing stability and efficiency in exploration-heavy environments. Specifically, [19] used a shared network with K distinct bootstrap heads, where a head corresponds to a number of layers (e.g. 2) added after the shared layers. Each head has its own target network and parameters, and at each episode a head is assigned at the beginning to be utilized throughout the whole episode. Additionally, randomly selected experiences are shared between the heads. In more detail, a masking vector m_i for a specific observation i, is a binary vector of length K that determines which heads will be trained in each experience tuple. Hence, the m_i^k bit of that mask determines whether the weights of a head k will be updated using observation i. By finetuning these parameters, Bootstrapped DQN arguably facilitates deep exploration.

The incorporation of multiple learning heads, makes the algorithm more appealing for multiagent settings, while allowing the sharing of information via the shared network layers. However, in our domain of interest, having the heads share experiences at random may raise privacy concerns regarding the potential exposure of sensitive information across multiple agents or groups. Additionally, selecting a single head to be utilized by all agents per episode may lead to further privacy-related vulnerabilities, as this makes the decision-making a single entity's reliability, compromising the autonomy of the agents. This is a limitation our work tackles, as we explain later in detail.

Now, coalition formation theory is based on cooperative game theory, as autonomous agents form coalitions to gain rewards [1]. However, the *sequential* aspects of the problem, related protocols, and learning algorithms have been the focus of multiagent systems research. Coalition formation is key for joint task execution [2], [18], requiring agents to allocate resources to teams, and sequence these allocations to maximize long-term rewards [3], [21]. Moreover, in the face of uncertainty, *learning* comes naturally into the picture.

The work of [3], [5], [22], for instance, presents a framework that employs reinforcement learning to achieve *sequentially optimal repeated coalition formation under uncertainty*. Their framework essentially intertwines repeated coalition formation leading to rewards for the agents with *Bayesian model-based RL* to learn the agents' capabilities or *types*. This approach enables agents to explicitly consider uncertainty when making decisions, leading to more robust and adaptive coalition formation strategies. However, that approach has yet to make the transition to the deep learning paradigm.

Indeed, all deep RL approaches that have been used for coalition formation to date are, to the best of our knowledge, model-free DRL ones. Most of that literature actually consists of rather straightforward applications of DQN to various domains calling for the formation of coalitions. For instance, in the Smart Grid domain, several approaches employ DQN to match sellers to buyers, or producers/consumers to energy aggregators [14], [15]. Regardless, the coalition formation process is rather simplistic: agents propose to become members of a coalition after selecting such a coalition randomly; "coalition leaders" approve such applications one at the time; and/or the process is entirely centralized. It is not therefore clear how the approaches appropriately address *the sequential aspect* of the autonomous agents' formation decisions problem in a meaningful manner. Departing from the Smart Grid domain, [16] proposes an *Imitation Augmented Deep Reinforcement Learning (IADRL)* model that enables an Unmanned Ground Vehicle and an Unmanned Aerial Vehicle

to form a coalition to perform tasks that they are incapable of achieving alone. The approach however is centralized, and limited to 2-agent coalitions. Then, Bachrach et al. [23] have used several off-the-shelf model-free DRL algorithms in order to train agents to *negotiate* and form teams that assign rewards to agents given their negotiated and agreed-upon demands, but only one coalition emerges as the result of negotiations in a given round. Thus, agents *do not strategize about contributing parts of their resources to various coalitions across rounds*, thus significantly simplifying their sequential decision-making problem; while their experiments involve only five learning agents in *fully observable* environments, and the reward function is non-stochastic and known to all participating agents.

Our work employs model-free DRL for coalition formation. In contrast to existing approaches, our framework allows agents to take *sequential formation decisions* in an autonomous, non-centralized manner; and increases learning efficiency via privacy-preserving information sharing. In a nutshell, our approach makes use of the notions of *agent types* and *social groups*, so as to enable *DRL as a tool for sequential coalition formation towards effective coalitional task execution* in *any* large open multiagent setting of interest.

III. Our Approach

We now present the multiagent environment, and proceed to define our DRL-enabled coalition formation framework.

The environment is populated by N agents, in the set $A = \{A_1, A_2, ... A_N\}$, $N \in \mathbb{N}$. Each agent A_i has a discrete quantity of resources denoted as $r_i \in \mathbb{N}$. Agent resources r_i are the basis for proposing and participating in coalitions $C \in \mathcal{C}$, which are sets of agents that perform a task, and where $\mathcal{C}$ denotes the space of all possible coalitions. The agents' strategic decisions of participating or not in a particular coalition, is a (strategic) resource allocation problem and it affects the results of the collective endeavor at hand.

Each agent A_i is characterized by its type $t_i \in \{T_1, T_2, ... T_M\}$, where $M \in \mathbb{N}$, and typically $|M| \ll |N|$. When this assists comprehension, we henceforth refer to agents along with their corresponding type, i.e., $A_i^{t_i}$. The *coalition type vector* encompassing the types of agents in a coalition C is denoted as $\mathbf{t}_C$. The type of each agent dictates the amount of resources that the agent contributes upon entering a coalition. Moreover, t_i also influences the social interactions within the environment. In realistic scenarios, agents may belong to parent entities,[2] or in teams and social networks that comprise sets of agents of various types. This means that each agent belongs to one of the $L \in \mathbb{N}$ *"social groups"* SG_l, $l \in \{1, ... L\}$; and each SG_l may include agents of different types t_i. Agents have knowledge only of their own types and the coalition-task pairs (and corresponding rewards) they have already participated in, and not of others' types, nor the task rewards to be gained from forthcoming coalitions.

Each task $\mathcal{T}_C \in \{\mathcal{T}_1, \mathcal{T}_2, ... \mathcal{T}_n\}$, $n \in \mathbb{N}$ is assigned to a particular coalition C, and is characterized by its requirements, i.e., a quota of resources $q(\mathcal{T}_C) \in \mathbb{N}$, that must be provided in order for it to be considered as completed. Note that, we consider $q(\mathcal{T}_C)$ to be also unknown to the agents, who are in fact tasked with determining over time the effectiveness of coalitions to complete tasks and gather task-specific rewards that actually depend on the coalition members' types. The resources of $q(\mathcal{T}_C)$ do not need to be accumulated by a single agent type, but from the corresponding coalition which comprises different types. Since analyzing the intricacies of resource allocation lies beyond the scope of this work, in contrast to existing literature [24], [25], we simplify the resource allocation process. Rather than requiring agents to gather multiple types of resources with assigned weights, we consider only a single resource type. What varies is the resource provision, which is governed by the agent types t_i. This simplification helps us focus on the RL and sequential aspects of the coalition formation process—i.e., allowing the agents to discover the underlying multiagent collaboration patterns among agent types. Nevertheless, generalizing to a setting with more than one resource type is straightforward as it is performed in the scope of the subtask that estimates agent types, and can be effectively tackled by our approach. The objective of the agents is to learn the reward function, so as to maximize reward by forming beneficial coalitions. This in turn naturally leads to the improvement of the overall reward accrued by all coalitions formed over time.

A. A Sequential Coalition Formation Protocol

Here we put forward a generic coalition formation protocol, spanning over a number of rounds, that allows agents to propose coalitions to execute tasks; and also to evaluate such proposals in order to accept or reject them.

Our protocol operates as follows.[3] At the beginning of a *coalition formation episode*, a number $n \in \mathbb{N}$ of tasks are issued. The episode consists of multiple rounds. At the first round of an episode, a proposer is selected randomly, and is then able to make proposals to form coalitions to complete tasks. Specifically, a proposal is a pair $\mathcal{P} = \{\mathbf{t_C}, \mathcal{T}_C\}$, where $\mathbf{t_C}$ is a vector specifying the particular agent types t_i for which the proposer agent is confident that suffice to form a coalition C to complete the task $\mathcal{T}_C$. The proposer makes one proposal per round. We assume that the number of proposals that the proposer is allowed to make is bound by a pre-defined limit $p \in \mathbb{N}$; and when p is reached, or when all the resources of the proposer have been allocated to formed coalitions, a new proposer is chosen among the agents that have not played that role before. Proposals require unanimous approval by the agents addressed to (the responders); otherwise, new ones are submitted. Responders either accept and contribute resources towards the task, with rewards distributed upon task completion and penalties for unmet requirements; or reject the proposal. A formation episode is terminated if either coalitions have formed for all tasks in offer; or if every agent has been the

[2]For instance, agents may belong to a parent organization or company participating via its subsidiaries in coalition formation activities.

[3]The complete protocol is provided in pseudocode form at https://anonymous.4open.science/r/Supplementary-material-C767/

proposer once, and no further coalitions can form due to either lack of agreement or depletion of available agent resources.

Now, an agent seeks to maximize rewards by suggesting an optimal sequence of proposals (coalition-task pairs) $\mathcal{P}$ when it is the proposer; and by responding to sequences of proposals optimally when being a responder. There is a sequential value of proposals and acceptance decisions agents face, given their finite resources that need to be optimally allocated to coalitions across the episodes' rounds in order for their rewards to be maximized. Of course, agents need to take these decisions in the face of uncertainty regarding coalitional values. (Deep) RL will, in principle, enable them to assess the sequential value of their decisions, via approximating the Bellman optimality equations [26]. Regarding the nature of these DRL algorithms, however, we note that large, open environments with sparse or scarce agent interactions, privacy constraints, and inherent stochasticity, prevent using algorithms that assume precise knowledge of coalition values or agent capabilities. Instead, we put forward privacy-aware DRL algorithms to enable sequential decisions in such demanding coalitional task execution settings. In effect, our framework helps make the transition from using tabular RL for coalition formation under uncertainty [3], [5], [22], [27] to the deep learning era.

B. Deep RL for Coalition Formation

We now present our privacy-aware DRL framework and algorithms. To begin, note that agents taking formation decisions utilize a DRL module that models the environment as an MDP [26]. This is equipped with an appropriate reward function.

Reward Function Design: In our work agents form coalitions under structural uncertainty about the value of their synergies, modeled by a function $v(\cdot)$ inspired by *marginal contribution nets (MC-nets)* [28] and *relational rules (RR)* [29].

The reward function is as in Eq. 1, and provides positive payoffs if the task resources quota $q(\mathcal{T}_C)$ is met or *penalties* if unmet. Reward is split equally among coalition members).

$$R(\mathbf{t}_C, \mathcal{T}_C) = \begin{cases} \sum_{\forall i,j \in C, i \neq j} v(\mathcal{T}_C, t_i, t_j), & q(\mathcal{T}_C) \text{ is met} \\ -\sum_{\forall i,j \in C, i \neq j} v(\mathcal{T}_C, t_i, t_j), & \text{otherwise} \end{cases} \tag{1}$$

In Eq. 1, v is a (characteristic) function that determines the effectiveness of collaboration or synergy among any pair of types t_i, t_j, that participate in a coalition to complete task $\mathcal{T}_C$. Moreover, we consider v to be stochastic, since *(i)* in real-world scenarios the value of any multiagent synergy cannot be considered as granted; and *(ii)* the introduction of stochasticity inside the reward function has been shown to encourage agent exploration and to make their learning more robust [30]. This uncertainty is inherent in the environment and not related to the partial observability due to privacy restrictions.

Stating a proposal: The proposer uses DRL to determine a coalition-task pair $\mathcal{P}$ to propose based on Q-values at hand. The set of all coalition-task pairs constitutes the action-space. The proposer balances exploration and exploitation in an ϵ-greedy fashion: A random action (proposal) $\mathcal{P}^-$ is chosen with probability ϵ; or the coalition-task pair $\mathcal{P}^*$ determined as best (given its Q-value) by the group's so far trained model is proposed. ϵ decays according to the rule $\epsilon = max(\epsilon_{min}, \epsilon \cdot \phi)$, where ϵ_{min} is the lowest value allowed for ϵ and ϕ a positive $(0 < \phi < 1)$ decay factor for ϵ.

Proposal Evaluation: A responder uses DRL to compute Q-values corresponding coalition-task pairs, and determine whether it can accept a $\mathcal{P}$ proposal. The set of all coalition-task pairs constitutes the action-space. $\mathcal{P}$ is accepted if its $q^{\mathcal{P}}$ Q-value exceeds an $(1-z)\bar{Q}_k$ threshold, where $\bar{Q}_k$ is the average of its top-k proposals' Q-values, and $z \in (0, 1)$:

$$q^{\mathcal{P}} \geq (1-z)\bar{Q}_k \tag{2}$$

During initial exploration, when Q-values are unreliable, responders conduct a secondary check. This uses the Jaccard similarity JS between the proposal's type vector t_C and the top-k type vectors. If JS exceeds a threshold the proposal is accepted; otherwise it is rejected. This approach balances robustness and early-stage exploration.

C. Mutliple-DQN Method

As argued earlier, adopting a shared network for learning is unsuitable in coalition formation settings with privacy concerns. Moreover, while independent DRL algorithms (e.g. DQN) could be used by individual agents, the sporadic coalition participation in open settings limits continuous learning opportunities.[4] Instead, agents grouped into "social groups" can learn collectively, even if group members differ in type t_i.

Our first proposed privacy-preserving algorithm we equip our DRL module with is *Multiple-DQN*. It simply employs a distinct DQN for each social group, trained solely on data from agents within the group. *This strict separation ensures privacy by design, and prevents inter-group interference, as no information or network layers are shared across groups.*

To elaborate, each DQN operates independently, leveraging the data collected from its social group to approximate the Q-values for coalition formation decisions. Each DQN's network architecture is identical, consisting of fully connected layers, but with weights trained solely based on the experiences of the respective group's agents. This independence allows the networks to tailor their learning to the specific dynamics and behaviors of their social group, without being influenced by unrelated or potentially disruptive data from other groups.

Although promoting privacy and group-specific learning, this method has limitations. The lack of shared knowledge between groups hinders the ability to leverage generalizable patterns, and groups with fewer data samples or active agents may face slower learning and reduced performance.

D. Privacy-Aware Multi-head DQN

Our second privacy-aware DRL method is *Privacy-Aware* Multi-Head DQN (PAMH-DQN), inspired by *Bootstrapped DQN* [19]. In *Bootstrapped DQN*, a shared network branches into K heads, with experiences shared among them. However,

[4]Notice that simulating and evaluating such a scenario with thousands of agents would be infeasible in practice due to computational requirements.

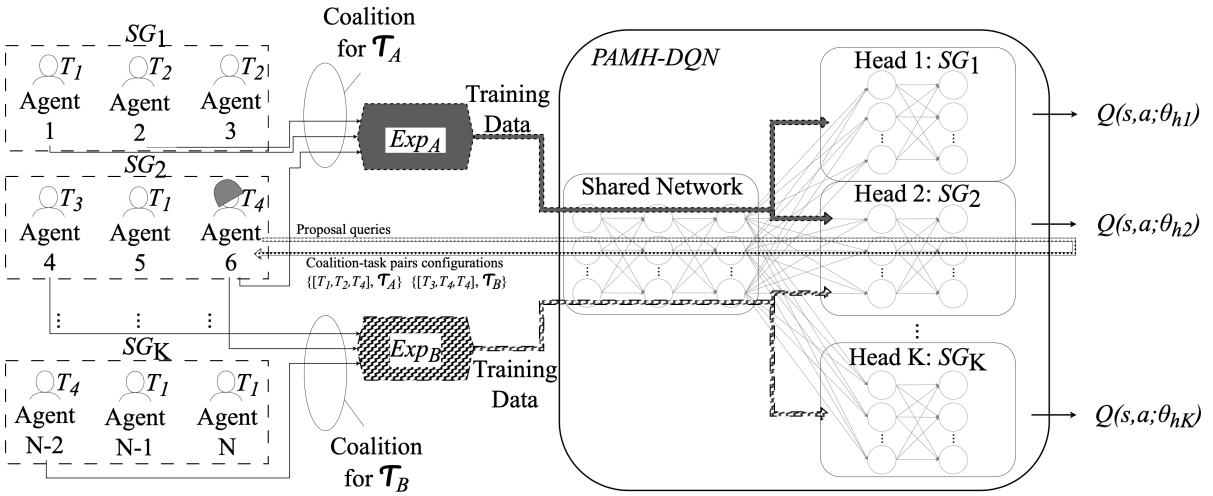

Fig. 1. An overview of the Privacy-Aware Multi-Head DQN architecture where each head/group consists of 2 fully connected linear layers and the common network which consists of 3 fully connected linear layers. The output represents the Q-values for the specific state-action pair, conditioned on the parameters of each head h, noted as θ_h. In this example, Head 1 and Head K do not share training data i.e. the results of past experiences Exp_A and Exp_A, since there has not been any task in which both SG_1 and SG_K members participated. The proposer (Agent 6) queries the network and retrieves coalition-task pairs configurations that presents as proposals. A responding agent would use the network in a similar fashion to decide whether to accept/reject a proposal.

in our approach, each head is dedicated to a social group: agents only access information relevant to their group while sharing anonymized summaries via shared network layers.

This gives rise to our PAMH-DQN architecture (Figure 1), in which each head corresponds to a particular *"social group"*. Unlike in Bootstrapped DQN, groups/heads *do not share information among them*, as each group has access only to the samples that are acquired by its members to train upon. Moreover, there is no need to select a different head for each episode as every agent can readily use only the head assigned to its group. Furthermore, for each head we compute a different loss considering only its own target network, which is then backpropagated throughout all the network layers, both the head-specific, and the shared ones. Inevitably, each head performing an update affects indirectly the other heads via the update of parameters of the shared network layers. For this reason, we choose to normalize the loss by the number of heads, i.e., we multiply each head's loss by $1/K$. Note that the shared layers' output does not represent reward estimates, but instead *encodings of the environment's states*. The output consists of Q-values, which reflect the immediate environment and the strategic implications of various future actions. The state of the environment, as perceived by a decision-making agent, consists of the amount of the agent's remaining resources. This value is crucial for guiding coalition formation behavior, since it dictates the number of coalitions the agent has the capacity to participate in during the upcoming rounds. This approach balances privacy and shared learning benefits without exposing sensitive information, similar to parameter sharing in [31]. [5]

[5]In real-world scenarios where a trusted third-party cannot be necessarily assumed, each social group could maintain an individual network, and exchange at frequent time intervals the values for the shared network's weights. This guarantees privacy preservation, as no sensitive data is exchanged among different groups, nor a trusted third-party is assumed.

Definition 1. *Sensitive information is the set of coalition task pairs $\mathcal{P}$ and the corresponding reward values R, $d^* = \{\mathcal{P}, R(\mathbf{t_C}, \mathcal{T}_C)\}$.*

This data is revealed only to the agents that participate in each coalition, and is used to update the weights of the *PAMH-DQN*'s shared layers and social group-specific heads.

Theorem 1. *PAMH-DQN does not compromise privacy among social groups.*

Proof. [See here for a full proof. Here we provide a sketch.]

Forward Pass: PAMH-DQN consists of a number of shared layers C and K heads; each head is utilized only by the corresponding social group. During the forward pass of the network, the neuron weights (both of the shared layers and the head-specific ones) are not updated and hence, sensitive information does not leak. In addition, during the forward pass between the shared network and the heads, the information is multiplied by the mask. In this way, the heads of different social groups do not have access to sensitive information of other groups and thus cannot perform any kind of calculations that use sensitive information of others.

Let $y = d^*[1] + \gamma max_a Q(s', a)$ be the target of the NN, and h_k^l is the output of the l layer of the k^{th} head. The output of a shared layer is defined as h^l_{shared}. Finally, we define L as the last layer of each head and M as the last layer of the shared network. Hence, $h_k^L, \forall k$ is the output of the last layer of each head and h^M_{shared} is the last layer of the shared network. We also define as $I_o^{(k)}$ the input to each head for that specific observation o; this input is derived from the last layer of the shared network. That is, $I^{(k)}$ can be written as:

$$I^{(k)} = m_o^k \cdot h^M_{\text{shared}}, \forall k$$

where m_o^k is the mask for observation o, i.e. a K dimension

one-hot vector that only the k^{th} bit is 1. Hence, during the forward pass, the information from that shared layer is only passed to the k^{th} head. Note that k may also represent an array, as a single experience can influence multiple heads/social groups. However, for notational simplicity, we will treat k as a scalar throughout the remainder of the .[6]

Backpropagation: During the backpropagation, each social group's head is trained on different data. That is, the gradients g_o^k calculated in the last layer of each head k are:

$$g_o^k = m_o^k \left(y - h_k^l\right) \nabla_{w_k} h_k^l.$$

Thus, during the backpropagation of sensitive data belonging to social group k, the gradient of all heads $i \neq k$ will be zero. This can be also seen from the formula for upgrading the weights of another head i:

$$\frac{\partial L}{\partial W_i^{(l)}} = \frac{\partial L}{\partial h_i^{(L)}} = \frac{\partial h_i^{(L)}}{\partial h_i^{(l)}} = \frac{\partial h_i^{(l)}}{\partial W_i^{(l)}} = 0, \forall i \neq k$$

Hence, in the case of backpropagating gradients generated by data from a social group k, only the weights of k^{th}head are updated; the other heads are not affected by that data in no means possible.

Updating the shared layers: We now consider the update of the shared layers. During the backpropagation step, the gradient for a *shared layer* l, is calculated as (chain rule):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{\text{shared}}^l} = \frac{\partial \mathcal{L}}{\partial h_k^L} \cdot \frac{\partial h_k^L}{\partial \mathbf{h}_{\text{shared}}^l} \cdot \frac{\partial \mathbf{h}_{\text{shared}}^l}{\partial \mathbf{W}_{\text{shared}}^l}$$

where $\mathbf{W}_{\text{shared}}^l$ are the weights of the shared layer l and $\mathcal{L}$ is the Huber loss calculated as:

$$\mathcal{L}(y, h_k^L) = \begin{cases} \frac{1}{2}(y - h_k^L)^2 & \text{if } |y - h_k^L| \leq \delta, \\ \delta\left(|y - h_k^L| - \frac{\delta}{2}\right) & \text{otherwise,} \end{cases}$$

The Huber loss $\mathcal{L}(y, h_k^L)$ yields gradients:

$$\frac{\partial \mathcal{L}}{\partial h_k^L} = \begin{cases} h_k^L - y \\ \delta \cdot \text{sgn}(y - h_k^L) \end{cases}$$

$$= \begin{cases} Q_l^L(s,a) - d^*[1] - \gamma \max_{a'} Q(s', a'), \text{ if } |y - h_k^L| \leq \delta \\ \delta \cdot \text{sgn}(d^*[1] + \gamma \max_{a'} Q(s', a') - Q_l^L(s,a)), \text{ otherwise} \end{cases}$$

where the $sgn(x) = -[x < 0] + [x > 0]$. Hence sgn is non-bijective, since different sgn inputs can be paired to the same sgn outputs—for instance, $sgn(+2) = sgn(+3) = +1$ and $sgn(-1) = sgn(-15) = -1$. *Notice this is enough to guarantee that* $\frac{\partial \mathcal{L}}{\partial h_k^L}$ *and therefore gradients* $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l}$ *are non-bijective.*

In addition, for interest, we note that the reward function (and thus the Q-function) is non-bijective since we guarantee in our implementation that our characteristic function v is both stochastic and non-injective. For example, assume $v(\mathcal{T}_C, A, B) = v(\mathcal{T}_{C'}, E, F) \sim \mathcal{N}(30, 20)$ where A,B,E,F are distinct types of agents participating in (two distinct) coalitions tackling tasks $\mathcal{T}_C$ and $\mathcal{T}_{C'}$; and $\mathcal{N}$ is a gaussian distribution with $\mu = 30$ and $\sigma^2 = 20$. Since it is possible that $v(\mathcal{T}_C, A, B) = v(\mathcal{T}_{C'}, E, F) = c$, we cannot infer which coalition-task pair $d^*[0]$ produces a reward equal to c.

Since gradients $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^l}$ are non-bijective and therefore unable to uniquely identify y or h_k^L, no sensitive information can be inferred from the weight updates (and hence the layers) of the shared network.

□

We note here that in our experimental evaluation we also use a straightforward *Shared-DQN* method as a baseline. The method essentially employs a DQN that is trained using all available agents' experiences. There is no masking of training data with regard to the involvement of each social group, and the entirety of the network layers is common. Thus, all social groups access the same layers, which are trained on all the transitions sampled. This violates privacy since all agents and all social groups have full access to a network trained directly on data that may not involve them. Though one could expect the use of all available information to assist learning, it is also conceivable that ensuing cross-agents and cross-group interference has negative effects on the learning process. This latter phenomenon was actually observed in our experiments.

IV. Experimental Evaluation

In this section we present our experimental evaluation.

Setup There are $N = 1000$ agents in our environment. Every 4 episodes 1.5% of the total agents may be removed from, or added to the set of available agents, thus simulating an open environment, where agents may arrive or depart in a dynamic manner. There are $M = 10$ types and $L = 6$ social groups; while we allow at most 10 agents in a coalition. Agent i's resources r_i are sampled from a uniform distribution $\mathcal{U}(50, 150)$. The same type of distribution is used to generate the types' contribution, but with range $\mathcal{U}(10, 40)$.

In our experiments we employ the protocol of Sec. III, in which $n = 25$ tasks are issued and await potential completion over one episode composed of multiple rounds. When selected to be a proposer, an agent is allowed to make a number of at most $p = 10$ proposals. The parameter z in Eq. 2 is set to 30%. We note that our framework can incorporate *any* formation protocol. For instance, we also implemented and tested an alternative protocol in which the proposer changes in every round (i.e., $p = 1$). Our results for that protocol were similar to the ones reported here, with variations of less than 1%. We also note that the action space consists of 4,618,625 possible coalition-task pairs. This is calculated as $n \sum_{k=2}^{M} \binom{M}{k}$ where k is the potential number of agents in a coalition.

Each layer in every network has 32 neurons. Our *PAMH-DQN* consists of 3 fully connected linear layers that comprise the shared network and 2 fully connected linear layers for each head/group. The hyperparameters of the three network

[6]In our implementation, during the forward pass, we directly connect the output of the shared layer to the input of the appropriate head k. Hence, we do not perform any computations in the non-appropriate heads

TABLE I
COUNTS OF COALITION-TASK PAIRS THAT EACH SOCIAL GROUP OBSERVES IN AN EXAMPLE SIMULATION RUN.

Social Groups	Social Group 1	Social Group 2	Social Group 3	Social Group 4	Social Group 5	Social Group 6	Total
Samples	588	553	580	570	596	601	3488

TABLE II
ALGORITHMS' PERFORMANCE. TOTAL ACCUMULATED REWARDS ARE AVERAGES OF 50 RUNS WITH DIFFERENT SEEDS ACROSS ALL AGENTS.

	Simple Learning				Transfer Learning			Transfer Learning (cont. training)		
Network	*Q-learning*	*Shared-DQN*	*Multiple-DQN*	*PAMH-DQN*	*Shared-DQN*	*Multiple-DQN*	*PAMH-DQN*	*Shared-DQN*	*Multiple-DQN*	*PAMH-DQN*
Avg. Total Acc. Reward	2,435,089.47	3,794,218.20	3,742,879.56	**3,838,281.94**	7,292,021.63	**7,558,112.64**	7,451,558.17	7,288,960.22	**7,512,232.49**	7,439,262.32
Std σ	**8,852.334**	330,980.50	414,104.64	230,334.79	612,616.52	**427,255.97**	565,350.56	595,627.67	607,040.78	**572,685.44**
Min Reward	2,415,211.54	2,460,142.58	2,347,239.87	**2,952,577.52**	4,833,020.71	5,801,915.03	**5,918,477.89**	5,028,931.05	3,954,080.84	**5,827,890.19**
Max Reward	2,448,766.23	**4,306,770.07**	4,180,689.21	4,231,163.89	7,874,859.71	7,947,155.30	**7,958,559.78**	7,915,898.23	**7,937,934.35**	7,908,374.86
Duration(sec)	**1.42**	302.90	506.89	491.96	**293.83**	488.88	492.72	**292.74**	484.85	508.99

TABLE III
PAIRED T-TESTS FOR *total acc. rewards* AND *disc.-per-episode total acc. rewards*. $p < 0.05$ DENOTES STATISTICAL SIGNIFICANCE (IN BOLD).

	Simple Learning		Transfer Learning		Transfer Learning (cont. training)	
Paired t-tests for...	p-value (Total Reward)	p-value (Discounted-per-episode Tot Rew)	p-value (Total Reward)	p-value (Discounted-per-episode Total)	p-value (Total Reward)	p-value (Discounted-per-episode Total)
Shared-DQN/Multiple-DQN	0.2290	0.2211	**8.8932e-06**	**6.1111e-06**	**0.0063**	**0.0022**
Multiple-DQN/PAMH-DQN	0.1125	0.0856	**0.0240**	**0.0339**	0.2279	0.2291
Shared-DQN/PAMH-DQN	0.3992	0.3491	**0.0101**	**0.0225**	**0.0076**	**0.0028**

architectures we employ have been fine-tuned by performing grid search in similar, but smaller scale experiments. As explained in Sec. III, different DRL algorithms are trained over different sets of agent experiences. In Table I we present the number of samples that members of each social group participated in an example trial; the non-privacy preserving *Shared-DQN* is trained using all 3488 samples, while the *Multiple-DQN*s are trained using only the samples for each corresponding social group. *PAMH-DQN* utilizes all samples for training, but each head uses only the social group-specific samples. Experiments were run on an Intel i7-13700 CPU, with 64GB of RAM and an RTX 4060Ti-16GB GPU. We use Python 3.9.18, with Pytorch v.2.2.1 and Adam optimizer with learning rate $\alpha = 0.001$. The DQN discount factor was 0.99.

We test our methods in three scenarios: *(i)* Simple Learning; *(ii)* Transfer Learning; and *(iii)* Transfer Learning with continual training. We explain the scenarios in detail below. For interest, in the Simple Learning scenario we also compare the performance of our DRL methods to that of a baseline simple Q-learning [32] one, implemented as in [29].

We employ a metric we have developed to evaluate sequential performance in this domain. Simply using what is in RL referred to as discounted total accumulated reward, which discounts rewards at each learning step, is not applicable in our case. This is because agents might not be able to form coalitions and gain rewards at each round. Additionally, the discount factor is reset at the beginning of each episode, thus it is difficult to obtain an overall impression regarding the evolution of the learning process. Therefore, we coin a novel *discounted-per-episode total accumulated reward* metric devised as follows. We apply a discount rate of δ^e to the reward $R(e)$ accumulated within each e-th episode, with e ranging from 1 to maximum number of episodes used, and progressively summing these values up to produce the $\sum_e \delta^e R(e)$ function graphs shown in our figures. This metric smooths out the reward lines easing readability. Additionally it provides an overview of the sequential-across-episodes performance of our DRL methodology. Note, however, that this metric *does not* accurately reflect the sequential performance of the agents within each learning episode. Devising a metric to more accurately capture sequential performance *simultaneously* within and across episodes, is interesting future work.

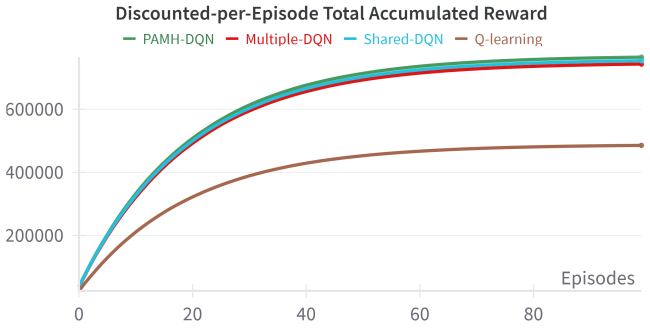

Fig. 2. *Simple Learning*: Discounted-per-episode Total Accumulated Reward, averaged over 50 runs. *PAMH-DQN*: $\sigma = 43,430.70$ *Shared-DQN*: $\sigma = 68,942.58$ *Multiple-DQN*: $\sigma = 81,998.66$. *Q-learning*: $\sigma = 2,828.59$

Simple Learning This scenario uses a reward function utilizing a v that draws values from a $\mathcal{U}(30, 20)$ distribution (cf. Eq. 1). Figure 2 depicts sequential performance, while "Simple Learning" section in Table II reports total (undiscounted) accumulated reward for all agents for 100 episodes.

All DRL methods clearly outperform the simple tabular *Q-learning*. Note also that the paired t-tests in Table III[7] show no statistical significant difference in performance among the DRL methods in this setting. However, *PAMH-DQN* ranks first in terms of *(a)* sequential performance (Fig. 2); *(b)* in terms of average rewards across all runs, and also *(c)* highest minimum rewards across all runs (i.e., the worst performing seed of *PAMH-DQN*, performs better than the worst performing seeds of both *Shared-DQN* and *Multiple-DQN*). *Shared-DQN* achieves higher *maximum* rewards across all runs. Moreover, it is important to remember that, unlike our methods, *Shared-DQN* does not respect any privacy concerns. On the other hand, *Multiple-DQN*, which trains only on experiences within its own social group, ranks third in all aspects in this scenario. This could be attributed to the method overlooking valuable general information that could be beneficial for its learning.

By contrast, *PAMH-DQN* manages to mitigate this problem by decoupling each head's learning process from others, allowing the learning of relevant (encoded) common knowledge while minimizing the impact of experiences not involving its assigned group. That is, each *PAMH-DQN* head learns only from experiences involving its social group, but also utilizes additional information regarding the environment dynamics—as encoded in the shared layers' output. The fact that it achieves the lowest standard deviation compared to *Shared-DQN* and *Multiple-DQN* underscores its stability.

Regarding time performance (Table II), *Q-learning* is the fastest due to its tabular update and no masking operations, while *Shared-DQN* uses a single network and no masking operations on the replay buffer experiences, whereas *Multiple-DQN* uses multiple networks, and *PAMH-DQN* combines shared and group-specific layers.

Transfer Learning Next, we review performance in an transfer learning scenario. Specifically, our models were trained using the reward function of the Simple Learning scenario, but evaluated in a setting with a different reward function. The latter uses a function v which draws values from $\mathcal{U}(60, 40)$.

Table II results show that *Multiple-DQN* ranks first, followed closely by *PAMH-DQN*. Moreover, Table II-"Transfer Learning" section, shows *Multiple-DQN* to surpass *PAMH-DQN*, even if not by a large margin, in terms of total average accumulated reward and standard deviation—as such demonstrating more stable behavior.However, *PAMH-DQN* is better in terms of highest minimum reward, maximum reward collected. Table III shows that both *Multiple-DQN* and *PAMH-DQN* perform statistically better than *Shared-DQN* in terms of both total average accumulated reward and discounted-per-episode total accumulated reward; whereas *Multiple-DQN* also outperforms *PAMH-DQN* in a statistically significant manner.

The inferior performance of *Shared-DQN* compared to that of *Multiple-DQN* and *PAMH-DQN* in the "Transfer Learning" scenario can be attributed to cross-group interference, where experiences from one group disrupt the learning process of other groups, even though all agents share the same network. Effectively, training on samples not associated with a particular group can be considered as noise in the training process of those groups that did not experience them. On the other hand, the fact that *Multiple-DQN* and *PAMH-DQN* train their group-specific heads exclusively or primarily on data of their actual social group, allows for more robust performance and better generalization and adaptability in the face of change.

[7]Wilcoxon tests we ran confirm the conclusions of the paired t-tests.

Transfer learning with continual training In this scenario, the models were originally trained for 100 episodes using the reward function of the Simple Learning scenario. Then, training was continued for a further 100 episodes with a reward function using function v which draws values from $\mathcal{U}(60, 40)$.

In the rightmost columns of Table II, we see that *Multiple-DQN* is slightly better than *PAMH-DQN* in terms of total accumulated reward; while *PAMH-DQN* has better stability, as it exhibits the lowest standard deviation compared to *Multiple-DQN* and *Shared-DQN*. Furthermore, both *Multiple-DQN* and *PAMH-DQN* exhibit statistically better performance than *Shared-DQN*, highlighting once again the negative impact of cross-agent and cross-group interference on the latter method (cf. Table III). Importantly, however, there is no statistical difference between our *Multiple-DQN* and *PAMH-DQN*. This is reassuring for the credibility of *PAMH-DQN*: the method is able to achieve results as good as those achieved by *Multiple-DQN* in this transfer learning scenario, while not requiring the maintenance of multiple networks as *Multiple-DQN* does. It appears that allowing *PAMH-DQN* to continue training with the new function also, an assumption which is actually realistic, enabled it to overcome any "cross-group interference" effects due to shared layers. Effectively, *PAMH-DQN*'s results confirm it as a method lying in the middle ground between *Shared-DQN* and *Multiple-DQN*, striking a good balance between respecting privacy and performance.

V. Conclusions and Future Work

In this work, we developed a framework for cooperative task execution in open multiagent settings, where agents take sequential decisions to form coalitions under uncertainty, sparsity of interactions, and privacy constraints. We put forward a generic formation protocol, and two novel DRL algorithms that exploit the concepts of social groups and agent types to allow the learning of coalition values and enable sequential decision making in such demanding settings. Our experiments demonstrate the effectiveness of our approach.

In ongoing and future work, we intend to experiment with alternative (possibly normalized) reward functions. Additionally, we intend to incorporate *type uncertainty* in our environment. This will make our learning more challenging, but also allow for increased transfer learning potential. Such an approach can also be paired with the incorporation of Graph Neural Networks (GNNs) [33] in our setting. GNNs bear the potential to effectively capture the structure of an environment involving multiagent interactions, and thus more accurately represent the relationships among agent types and also those of the agents within their coalitions and/or among social groups.

ACKNOWLEDGMENTS

The research described in this paper was carried out within the framework of the National Recovery and Resilience Plan Greece 2.0, funded by the European Union - NextGenerationEU (Implementation Body: HFRI. Project name: DEEPREBAYES. HFRI Project Number 15430).

REFERENCES

[1] G. Chalkiadakis et al., *Computational Aspects of Cooperative Game Theory*. Morgan & Claypool Publishers, 2011.
[2] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artificial Intelligence*, vol. 101, no. 1, 1998.
[3] G. Chalkiadakis and C. Boutilier, "Sequentially optimal repeated coalition formation under uncertainty," *Autonomous Agents and Multi-Agent Systems*, vol. 24, pp. 441–484, 2012.
[4] D. Ye, M. Zhang, and D. Sutanto, "Self-adaptation-based dynamic coalition formation in a distributed agent network: A mechanism and a brief survey," *IEEE Trans. on Parallel and Distributed Systems*, 2013.
[5] G. Chalkiadakis and C. Boutilier, "Bayesian reinforcement learning for coalition formation under uncertainty," in *AAMAS-04*, 2004.
[6] M. Moafi et al., "Optimal coalition formation and maximum profit allocation for distributed energy resources in smart grids based on cooperative game theory," *IJEPES*, vol. 144, p. 108492, 2023.
[7] F. Basso et al., "Coalition formation in collaborative production and transportation with competing firms," *Eur. J. of Operat. Research*, 2021.
[8] L. Vig and J. A. Adams, "Multi-robot coalition formation," *IEEE transactions on robotics*, vol. 22, pp. 637–649, 2006.
[9] S. Sarkar et al., "A survey on applications of coalition formation in multi-agent systems," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 11, p. e6876, 2022.
[10] M. Moafi et al., "Optimal coalition formation and maximum profit allocation for distributed energy resources in smart grids based on cooperative game theory," *Electr. Power & Energy Systems 144*, 2023.
[11] T. Wolff and A. Nieße, "Dynamic overlapping coalition formation in electricity markets: An extended formal model," *Energies*, 2023.
[12] Y. Liu et al., "Channel access optimization in unlicensed spectrum for downlink urllc: Centralized and Federated DRL approaches," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 7, 2023.
[13] R. Chen et al., "A DRL-based hierarchical game for physical layer security with dynamic trilateral coalitions," in *ICC 2023*, 2023.
[14] S. Orfanoudakis and G. Chalkiadakis, "A novel aggregation framework for the efficient integration of distributed energy resources in the smart grid," in *Proceedings of AAMAS-23*, 2023.
[15] M. Sadeghi and M. Erol-Kantarci, "Deep RL based coalition formation for energy trading in smart grid," in *IEEE 5GWF*, 2021.
[16] J. Zhang et al., "Iadrl: Imitation augmented deep reinforcement learning enabled ugv-uav coalition for tasking in complex environments," *IEEE Access*, vol. 8, pp. 102 335–102 347, 2020.
[17] G. Xu et al., "Privacy-preserving federated deep learning with irregular users," *IEEE Trans. on Dependable & Secure Computing*, 2022.
[18] S. Kraus, O. Shehory, and G. Taase, "The advantages of compromising in coalition formation with incomplete information," 02 2004.
[19] I. Osband et al., "Deep exploration via Bootstrapped DQN," in *NIPS-16*.
[20] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[21] G. Chalkiadakis et al., "Coalition formation under uncertainty: Bargaining equilibria and the bayesian core stability concept," in *AAMAS-07*.
[22] G. Chalkiadakis and C. Boutilier, "Sequential decision making in repeated coalition formation under uncertainty," in *AAMAS-2008*, 2008.
[23] Y. Bachrach et al., "Negotiating team formation using deep reinforcement learning," *Artificial Intelligence*, vol. 288, 2020.
[24] E. Elkind, G. Chalkiadakis, and N. R. Jennings, "Coalition structures in weighted voting games," in *ECAI-08*. IOS Press, 2008.
[25] N. Qi et al., "A task-driven sequential overlapping coalition formation game for resource allocation in heterogeneous UAV networks," *IEEE Trans on Mobile Computing*, vol. 22, no. 8, 2023.
[26] R. Sutton and A. Barto, *Reinforcement learning: An introduction*, 2018.
[27] G. Chalkiadakis, "A Bayesian Approach to Multiagent Reinforcement Learning and Coalition Formation under Uncertainty," PhD thesis, University of Toronto, Toronto, ON, Canada, 2007.
[28] S. Ieong and Y. Shoham, "Marginal contribution nets: a compact representation scheme for coalitional games," in *ACM EC-2005*, 2005.
[29] M. Mamakos and G. Chalkiadakis, "Overlapping coalition formation via Probabilistic Topic Modeling," in *AAMAS '18*, 2018.
[30] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," in *AAMAS '18*, 2018.
[31] J. Qi et al., "Federated reinforcement learning: Techniques, applications, and open challenges," *ArXiv*, vol. abs/2108.11887, 2021.
[32] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, 1992.
[33] Z. Wu et al., "A comprehensive survey on graph neural networks," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, no. 1, 2021.

VI. APPENDIX

A. Coalition Formation Protocol

Algorithm 1 A DRL-enabling coalition formation protocol

Require: Initialized set of agents, A
Initialize the list of Tasks
for each episode **do**
  Reset the environment
  while no terminal condition has been met **do**
    An agent i is randomly selected as the proposer
    while proposer has made $< p$ proposals **do**
      Proposer i uses DRL to make a proposal
      Each responder agent uses DRL to respond
      if proposal is accepted **then**
        Coalition C is formed to complete task $\mathcal{T}_C$.
        The reward/penalty R for completing $\mathcal{T}_C$ is distributed equally among the participants.
      else
        Another proposal is made from proposer i.
      end if
    end while
  end while
end for

B. Responding agent's logic

Algorithm 2 Responder's Proposal Evaluation Protocol

An agent receives a proposal from the proposer
Uses its *"social group"*'s DRL network to generate the Q-values for the available actions of the responder
The responder calculates the mean of the top-k Q-values and compares it to the Q-value of the proposal it received
if proposal's Q-value is within threshold **then**
  Responder accepts the proposal
else
  With probability ϵ decaying over time: checks coalition type vectors similarity
  if similarity within coalition threshold **then**
    Accepts the proposal
  else
    Rejects the proposal
  end if
end if

C. Experiment Figures

D. Transfer Learning Scenario

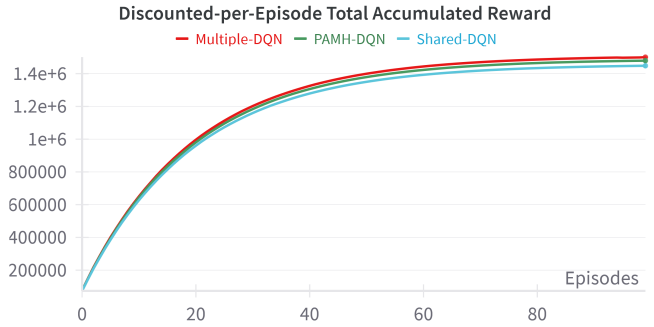

Fig. 3. *Transfer Learning*: Discounted-per-episode Total Accumulated Reward (avg. over 50 runs) in an environment with a new reward function, albeit with models trained on a previous one. *PAMH-DQN*: $\sigma = 116,458.41$; *Shared-DQN*: $\sigma = 126,056.58$; *Multiple-DQN*: $\sigma = 86,504.60$.

E. Transfer Learning with Cont. Training Scenario

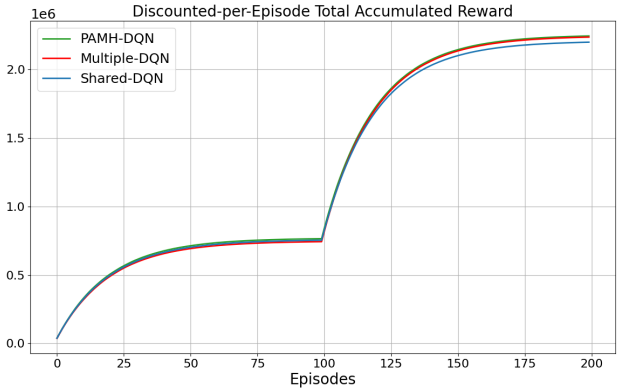

Fig. 4. *Transfer Learning with Continual Training*: Discounted-per-episode Total Accumulated Reward (50 runs average). In episodes 101-200 the model continues to train on a different reward function than the one used during episodes 1-100. *PAMH-DQN*: $\sigma = 115,391.90$; *Shared-DQN*: $\sigma = 122,294.50$; *Multiple-DQN*: $\sigma = 119,876.72$.