- Agents have types
- They propose a DQN with 2 heads and BARGAINING for the teams
- The reward is in Eq 1 - I believe they give the Marginal Contribution

Deep Reinforcement Learning for Coalition Formation in Large Open Environments

Paper #1516

Abstract. Task execution in multiagent environments often calls for the formation of coalitions of individual agents, since these may possess complementary skills and/or resources that have to be pulled together for a task. Naturally, the coalition formation process and its efficiency is affected by the strategic choices of the participating agents, as well as by the uncertainty inherent in practically all real-world settings; and thus it can in principle be facilitated by reinforcement learning (RL). The practicality of employing RL by each individual to learn aspects of the coalition formation problem in large open settings is questionable, however. As such, in this work we propose a novel, *deep* RL-based paradigm that enables agents to learn the values of potential coalitions in such challenging environments. This is facilitated by the assumption of capability-related agent types; and by allowing the joint RL training of agents whenever this does not jeopardise their privacy concerns, via a *privacy-aware multi-head DQN* algorithm we put forward. To the best of our knowledge, our approach is the first to allow autonomous agents to employ deep RL in large open sequential coalitional task allocation settings; while addressing scalability and privacy concerns, as well as the uncertainty related to the reward function. Our experimental results verify the effectiveness of our approach; and demonstrate the potential for transfer learning in this setting, as an already trained model can be successfully employed to environments with different reward functions.

1 Introduction

Coalition formation is a microeconomics-originating paradigm widely used in the dynamic landscape of multiagent systems. It is used to facilitate the coordination of autonomous agents and the utilization of their resources in order to complete joint tasks [7, 29]. In general, coalition formation faces challenges regarding *(i)* scalability, when the number of agents is large [20], because the number of coalitions we have to consider grows rapidly; *(ii)* increased models' complexity when dealing with evolving and dynamic environments [4, 40], since any learning that has been achieved becomes obsolete when agents enter or leave or any component of the coalition formation process changes; and *(iii)* communication constraints and restrictions regarding the information agents are allowed to exchange [39].

Moreover, especially in dynamic multiagent settings, the efficiency of the formed coalitions is often threatened by inherent uncertainty regarding aspects of the agents or the environment, such as the agent skills or types, the contribution capabilities, the difference between expected and realized utility, etc [2, 3, 4]. Whenever attempting to form coalitions under various forms of uncertainty, it is imperative that agents acquire the means for *learning* the values of their (prospective) coalitions, and/or the capabilities of their peers [2, 4, 20, 23, 29, 35]. In large open multiagent coalitional task allocation environments, however, individual learning arguably becomes impractical due to the fact that *(a)* specific agents may not be encountered more than a few times in a given agent's interactions with the system; and, *(b)* in general, reduced learning opportunities come up due to potentially large periods of inactivity—e.g., when an agent is not selected by others to contribute to tasks. Both issues result to rare training instances, and thus the learned models do not fit adequately.

An answer to the first issue could be to employ a form of "transfer learning"—e.g., tailored to learn the peer agents' *types* that correspond to capabilities or skills that could be used in the context of various coalitions or tasks, as proposed by past work that employs (model-based) Bayesian RL [2, 3, 4]. Although insightful, such approaches do not tackle the second issue regarding sparse participation opportunities; while their tabular nature raises scalability barriers.

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 [15, 18] involve a centralized 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 [8, 24, 28, 31, 41] but they also raise challenges related to scalability and communication overhead.

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, in many cases introduces important privacy concerns—since, e.g., a shared network infrastructure could be exploited for scavenging personal data, such as experiences of other agents [38]. 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 very 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 [16, 29]); 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 that belong to the same *"social group"* to utilize shared neural networks, or just some layers of such, to enhance their learning process and increase its efficiency.

Against this background, in this work we propose a deep RL methodology—encompassing a novel model-free DRL algorithm and an appropriate accompanying coalition formation protocol—for learning the value of agent coalitions in large sequential task allocation environments, where agents share their available resources among different coalitions. Importantly, our proposed algorithm respects agents' privacy concerns via employing a variation in the logic of *multi-head DRL* techniques. Specifically, inspired by other works that utilize shared layers of neural networks [17, 25], we modify the classic DQN algorithm [22] 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 belonging in distinct social groups, since we ensure that each head is *dedicated* to its social group—i.e., it has access to experiences attained by the members of its associated social group only.

Summarizing, there are four main contributions of our work in this paper. First, we propose a novel framework for the deep reinforcement learning of coalition values within environments characterized by diverse agent types in large open environments. Second, we introduce a social network perspective to coalition value learning, respecting the privacy concerns regarding the information that is shared and communicated among the agents and the coalitions in such environments. Third, we technically achieve this via a novel use of the heads of the neural network employed by our proposed *PAMH-DQN* DRL technique. Fourth, our thorough experimental evaluation verifies the effectiveness of our *PAMH-DQN* approach for learning in large open coalitional task allocation environments; while it demonstrates its potential for allowing already trained models to be used in environments with different coalitional reward functions.

The rest of this paper is structured as follows. Section 2 lays out the theoretical foundations for our work and reviews related literature. Section 3 elaborates on the inner workings of our model and algorithm. Section 4 presents our experimental evaluation and results; while Section 5 concludes and outlines future work directions.

2 Background and Related Work

Reinforcement learning (RL) is a powerful machine learning paradigm enabling the training of agents via trial-and-error, in order to take long-term reward maximizing sequential decisions [30]. Perhaps the most well-known RL algorithm is Q-learning [34]. Q-learning can handle problems with stochastic transitions and rewards without resorting to a model of the environment. Rather, it focuses on estimating the Q-function, which is used to represent its expected cumulative reward, via the immediate reward "samples" it collects via acting in the environment [30]. Like all "tabular" RL approaches, however, Q-learning faces limitations particularly in dynamic and high-dimensional environments. First of all, it requires observing the effect of a large number of actions and set of probable states, whose size can grow exponentially in real-world settings. Additionally, Q-learning can suffer from convergence issues in environments with continuous states spaces or sparse rewards, making it difficult to find optimal policies efficiently [30]. To address these limitations, Deep Q-Networks (DQN) and its extensions [21, 22, 32] were introduced, taking advantage of the power of neural networks to approximate the Q-function, managing to handle complex and high-dimensional state-action spaces, and thus kicking-off the *deep reinforcement learning* era.

One of the major innovations that DQN introduced is the so-called *experience replay*: past experiences are reused multiple times to diminish temporal correlations between consecutive experiences, and in this way the stability of the learning process is enhanced. DQN also employs a target network to address challenges related to the target value estimation during training. By transferring the weights of the main network to the target network, the encoded knowledge is also shared among them. This mechanism mitigates the problem of fluctuating target values during training and prevents undesirable tribulations [10].

Building on DQN, *Bootstrapped DQN* [25] incorporates the concept of ensemble learning into its framework, further enhancing stability and efficiency in exploration-heavy environments. Specifically, [25] uses a shared network with K distinct bootstrap heads, where a head corresponds to a number of layers at the end of the network, e.g. the 2 last 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; and this approach arguably facilitates deep exploration.

The incorporation of multiple learning heads, as *Bootstrapped DQN* or other multi-head DRL variants do, makes the algorithm more appealing for multiagent settings, since such approaches facilitate the learning of multiple agents, while also allowing the sharing of information via the shared network layers. However, having the heads share experiences at random may raise privacy concerns regarding the potential exposure of sensitive information across multiple agents or groups. Additionally, the multi-head DRL practice of selecting a single head to be utilized by all entities (i.e. agents or social groups) per episode may lead to further privacy-related vulnerabilities, as this makes the decision-making a single entity reliability, compromising this way the autonomy of the agents. By contrast, we propose a different approach for utilizing multi-head DRL for the purpose of coalition formation, as the head that it is enabled each time is determined by the particular agent that utilizes it and the social group it belongs to. In this way we ensure that privacy is preserved among the social groups, as the experiences of agents in the same social group are only accessed during the training of the corresponding head.

Now, the theoretical foundations of coalition formation are laid mostly by cooperative game theory, since autonomous agents forming coalitions to collect joint team reward still engage in such activities for their own benefits [7]. However, the *sequential* aspects of coalition formation activities, related protocols, and learning algorithms have been the focus of work conducted in the multiagent systems community [36]. Coalition formation is key for task execution that requires the formation of teams [29, 16]; and agents possessing a given amount of resources face the natural challenge to choose how much of this amount to allocate to which teams, as well as to decide on the sequence with which to perform such an allocation, in order to maximize long-term reward [2, 3, 4, 5]. Moreover, in the face of uncertainty, reinforcement learning comes naturally into the picture.

The work of [2, 3, 4], 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*. The approach enables agents to explicitly consider uncertainty when making decisions, leading to more robust and adaptive coalition formation strategies. Model-based learning ideas can extend to *overlapping coalition formation* settings [6, 19, 20], in which agents can belong to multiple coalitions simultaneously. However, these coalition formation approaches have 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, [28] proposes the use of DQN for matching sellers to buyers in Smart Grid energy trading scenarios. Regardless, the coalition formation process is rather ad hoc: agents propose to become members of a coalition after selecting such a coalition randomly; and "coalition leaders" approve such applications one at the time. It is not therefore clear how the approach appropriately addresses the sequential aspect of the autonomous agents' formation decisions problem in a meaningful manner. Also applied in the Smart Grid domain, [31] uses deep RL for enabling consumers to sell/purchase a portion of their excess or lack of energy to others via transactive energy markets; again, coalition members offload the learning task to a coalitional representative. The approach of [24] also employs DQN to assign agents to energy aggregators in a Smart Grid setting, but does so in a centralized manner (i.e., aggregators run DQN to assess the impact that the addition of specific agents to their pool of energy assets would have to their demand-response capacity).

Departing from the Smart Grid domain, [41] 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 considers only 2-agent coalitions. Finally, Bachrach et al. [1] have recently used several off-the-shelf model-free DRL algorithms in order to train agents to *negotiate* and form teams that assign reward to agents given their negotiated and agreed-upon demands. In their domains, however, only one coalition emerges as the result of negotiations in a given round. Thus agents do not have to 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 environments that are fully observable and the reward function is non-stochastic and known to all participating agents.

In our work in this paper, we also employ model-free DRL for coalition formation. In contrast to most of the approaches mentioned, however, our framework allows the agents to take decisions in an autonomous, non-centralized manner; while allowing for more efficient learning via information sharing enabled by our novel, privacy aware, multi-head DRL algorithm that takes advantage of the notions of agent types and social groups to enable DRL for *sequential* formation decisions in large open coalitional task allocation settings.

3 Our Model

In this section, we elaborate on the setting of the multiagent environment we adopt, and analyze the DRL-based algorithm employed by our framework for sequential coalition formation.

3.1 Basics

We consider the environment to be populated by N agents in the set $A = \{A_1, A_2, ..., A_N\}$, $N \in \mathbb{N}$. Each agent A_i possesses a discrete quantity of resources denoted as $r_i \in \mathbb{N}$. Agent resources r_i are pivotal in proposing and participating in coalitions $C \in \mathcal{C}$, where $\mathcal{C}$ denotes the space of all possible coalitions. The agents' strategic decisions of whether to participate or not in a particular coalition is a resource allocation problem and it affects the results of the collective endeavour 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 $|M| << |N|$. 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 $\mathbf{t}_C$.

Now, the type of each agent dictates the amount of resources that the agent contributes upon entering a coalition, and this amount is noted as $c(A_i^{t_i})$. Moreover, t_i also influences the social interactions within the environment. In realistic scenarios, agents may belong to parent entities,[1] 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; and each SG_l may include agents of different types t_i.

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 $q(\mathcal{T}_C)$ that must be provided in order for it to be considered as completed. Note that in our approach $q(\mathcal{T}_C)$ is not known to the agents, who are in fact tasked with determining over time the effectiveness of coalitions to complete tasks and gather task-specific reward that actually depends on the coalition members' types. Now, since analyzing the intricacies of resource allocation lies beyond the scope of this work, in contrast to existing literature [9, 27], 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 kind of simplification helps us focus on the RL and sequential aspects of the coalition formation process—i.e., on allowing the agents to discover the underlying multiagent collaboration patterns among agent types.

3.2 A Sequential Coalition Formation Protocol

In each episode of our setting, the role of the proposer is assigned to an agent, who undertakes to generate and communicate particular coalition proposals to the other agents, for completing the required tasks. The receivers of the proposals either decide to take part in a coalition, or to refuse to participate and negotiate some other alternative. The agents that decide to provide resources for a coalition's task proceed to do so, and then they receive a reward at the successful completion of a task.

In more detail, the protocol that we assume is given in Algorithm 1. First, a list of tasks is generated that should be completed (line 1). Then, a number of episodes take place. At the beginning of each episode the parameters of the environment are reconfigured, and in particular the agent reward functions, the proposal counter, and the task completion status (line 3). An episode consists of a number of rounds that occur in an iterative manner until a terminal condition is met (line 4). At the beginning of each round an agent is selected randomly to be the proposer (line 5). This agent takes over the pivotal role of suggesting coalition formation proposals to the other agents with the aim to complete the given tasks.

Formally, 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 complete the task $\mathcal{T}_C$. It is important to note, that in order for a proposal to be considered accepted it needs to have been approved by every involved member, otherwise the proposer must submit a different proposal (line 12).

Now, by employing a DRL algorithm, the proposer seeks to optimize the suggested $\mathcal{P}$ pairs, so that the achieved rewards are maximized by as most successful task completions as possible.

[1] For instance, as mentioned in the introduction, agents either belong to a parent organization or company that participates via its subsidiaries in coalition formation activities.

We note that in this setting, agents can incorporate the use of any DRL module of choice. However, in our work we put forward a *privacy-aware multi-head DQN (PAMH-DQN)* algorithm which we believe is most appropriate for our problem at hand, for the reasons we explain later in Section 3.4. Also, the amount of proposals that the proposer is allowed to make is bound by a pre-defined proposal limit, denoted as p, where $p \in \mathbb{N}$. To that end, the proposer agent uses a DRL (i.e., in our case the PAMH-DQN) network with an $\epsilon - greedy$ training strategy in order to decide which action to take, i.e. which particular set of coalition-task pairs $\mathbf{P} = \{\mathcal{P}_1, ..., \mathcal{P}_p\}$ to suggest to the other agents (line 6). The $\epsilon - greedy$ is an algorithm that can balance exploration and exploitation. According to it, as the action to perform $\hat{\mathbf{P}}$, the proposer either chooses a random, possibly sub-optimal, action $\mathbf{P}^-$ with probability ϵ, or the best one as highlighted by their group's so far trained model, noted as $\mathbf{P}^*$, with $(1 - \epsilon)$:

$$\hat{\mathbf{P}} = \begin{cases} \mathbf{P}^* & \text{, with probability } 1 - \epsilon \\ \mathbf{P}^- \sim \mathcal{U} & \text{, with probability } \epsilon \end{cases}$$

The probability parameter ϵ decays as the algorithm progresses, so that later on we explore less and rely more on exploitation. After a proposal is accepted, the parameter decays, as follows: $\epsilon = max(\epsilon_{min}, \epsilon \cdot \phi)$, where ϵ_{min} is the lower limit that we allow the parameter to drop to and a decay factor ϕ, which is a positive number ($0 < \phi < 1$) that controls the rate at which ϵ drops.

Algorithm 1 A DRL-enabling coalition formation protocol

```
Require: Initialized set of agents, $A$
 1: Initialize the list of Tasks
 2: for each episode do
 3:     Reset the environment
 4:     while no terminal condition has been met do
 5:         An agent $i$ is randomly selected as the proposer
 6:         Proposer $i$ uses DRL to make $p$ proposals
 7:         Each responder agent uses DRL to respond
 8:         if proposal is accepted then
 9:             Coalition $C$ is formed to complete task $\mathcal{T}_C$
10:             The reward/penalty $R$ for completing $\mathcal{T}_C$ is distributed equally among the participating agents.
11:         else
12:             Another proposal is made from proposer $i$
```

After every participating agent accepts, the coalition is formed and the task is executed (line 9). Finally, the rewards are distributed to the contributing agents (line 10). Failing to meet the task's requirements—e.g. if agents wrongly estimated that a coalition would gather the required resources but finally did not—will lead to the members of the coalition receiving a penalty, specified as the negative amount of payoff generated from the task. As mentioned, in our setting we assume that each task has an (unknown to the agents) resource quota requirement $\mathcal{T}_C$—one that a coalition has to gather in order to successfully complete it.

The terminal conditions, mentioned in line 4 of Alg. 1, are another important element, which mark a state of our environment where the coalition formation process cannot continue. Such a condition e.g., is the case when all the generated tasks have been successfully completed. Also, having every agent become a proposer at most once marks a terminal condition in our case. Another condition is when the environment's agents run out of, or have insufficient resources to their disposal, in which cases agents are unable to form a coalition. Lastly, a case where the coalition formation has to be terminated is when we reach the maximum number of episodes, as set by the end-user.

Stating a proposal: The proposer uses DRL to determine the set of the most valuable coalition-task pairs $\mathbf{P}$ given their Q-value.

Proposal Evaluation: Upon receiving a coalition proposal, the responding coalition members initiate their own DRL algorithm (line 7 of Alg. 1). This enables them to evaluate the potential benefits of the proposed coalition in relation to their strategic objectives of maximizing their collective reward. Responders use a DRL network to generate the Q-values for their available actions at their state. The proposal evaluation procedure is shown in Algorithm 2. This is initiated each time a responder agent receives a proposal (line 1). Then, by utilizing the DRL the responder examines the top-k actions, where k is the number of coalitions the agent has the capacity to participate into, and compares the Q-value regarding the proposal the responder received with the average of the top-k actions' Q-values (lines 2-3). If the proposal's Q-value q^P lies over a percentage threshold z_q of the *average* of the top-k Q-values, $\bar{Q}_k$, i.e.,

$$q^P \geq (1 - z_q)\bar{Q}_k$$

then the responder accepts the proposal (line 5). Now, if q^P does not lie within this range, the responder would be inclined to decline. However, in the initial learning phases, where agents have not gathered much information regarding coalitional values, such a drastic approach could lead to the refusal of most proposals, since the actions' space (i.e., the space of possible coalitions) is huge and the agents' estimates regarding their values could vary drastically.

As such, we introduce an initial "exploration" phase whose duration is determined dynamically (Algorithm 2, line 7). In more detail, to balance exploration with exploitation from the responder agents' perspective, we use the decaying ϵ parameter in a similar manner to what we described earlier in 3.2, but this time instead of selecting a sub-optimal action with a probability ϵ, we initiate a secondary check. This is introduced to tackle the issue of agents repeatedly rejecting proposals as sub-optimal—e.g. during the first rounds of the training process when the Q-value is not being approximated accurately. Specifically, if the responder's assessment of the proposal's Q-value q^P is that it lies below the percentage threshold z_q, then with a (time-decreasing) probability ϵ, a secondary check is initiated. In essence, this secondary check is utilized to assess the similarity between the proposal's coalition structure and that of the top-k actions. It calculates the Jaccard similarity coefficient for the top-k actions' type vectors (instead of their Q-values as in the proposer's case). This similarity coefficient measures the degree of differentiation of the elements among two sets, in our case the two type vectors, that of the proposal and of the top-k action. If this coefficient lies within the percentage threshold z_q even for a single of the top-k actions, then the responder accepts the proposal (Alg. 2, line 9), otherwise the proposal is rejected (Alg. 2, line 11).

3.3 Deep RL for Coalition Formation

We assume that each agent utilizes a DRL module that is shared among all the members of its social group, and it models the environment as a Markov Decision Process (MDP) [30].

Reward Function Design: In DRL, to receive feedback regarding the suitability of an action chosen, agents need to acquire a reward dictated by a reward function $R(\cdot)$. The choice of a reward function is crucial for the discovery of appropriate agent policies. In this section, we detail our approach for designing such a function for our domain.

Algorithm 2 Responder's Proposal Evaluation Protocol

```
1:  An agent receives a proposal from the proposer
2:  Uses its group's DRL network to generate the Q-values for the
    available actions of the responder
3:  The responder calculates the mean of the top-k Q-values and
    compares it to the Q-value of the proposal it received
4:  **if** proposal's Q-value is within threshold **then**
5:      Responder accepts the proposal
6:  **else**
7:      With probability $\epsilon$ decaying over time: checks coalition type
        vectors similarity
8:      **if** similarity within coalition threshold **then**
9:          Accepts the proposal
10:     **else**
11:         Rejects the proposal
```

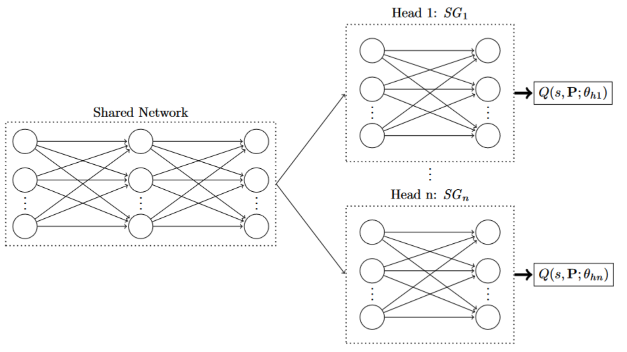
Head 1: SG_1
Shared Network
Head n: SG_n
$Q(s, \mathbf{P}; \theta_{h1})$
$Q(s, \mathbf{P}; \theta_{hn})$

Figure 1. Our Privacy-Aware Multi-Head DQN architecture where each head/group consists of two fully connected linear layers and the common network which consists of three 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 our setting, the state s is represented by the amount of available resources, and $\mathbf{P}$ represents all possible coalition pairs.

To begin, recall that the agents in our setting are considered to be fully autonomous with respect to their decisions regarding forming coalitions. Moreover, since agents *(i)* are not aware of the value that coalitions with given type vectors can receive for a task, and since *(ii)* they have the opportunity to progressively assign their resources to several coalitions within an episode, they are effectively obliged to *strategically* form coalitions under conditions of *structural uncertainty*. This corresponds to uncertainty surrounding the value of synergies among agents. In general, such a value can be seen as deriving from a characteristic function, $v(\cdot)$, determining the effectiveness and/or efficiency of collaboration among a group of agents. In principle, v can have any form deemed appropriate. In our work in this paper, to model the value of such synergies, and the way they affect coalitional reward, we drew inspiration from *marginal contribution nets (MC-nets)* [14] a concept which dictates how the value collaboration patterns affects the value of their coalition, and by *relational rules (RR)* [20], the MC-nets extension to overlapping coalition environments. We detail our approach immediately below.

With these in mind, we propose a reward function that evaluates a coalition-task assignment by taking into account the value of the synergies of pairs of agent types. In the case where the coalition fulfills the task's resource quota $q(\mathcal{T}_C)$, the member agents collectively receive a positive payoff, which is subsequently divided equally among them. In the case where $q(\mathcal{T}_C)$ is not achieved, a negative payoff is granted as a penalty. Our reward function is thus as in Eq. 1:

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

where, v is a (characteristic) function that determines the effectiveness of collaboration or synergy among any pair of types t_i, t_j participating 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 [12].

3.4 Privacy-Aware Multi-head DQN

In an environment where autonomous agents need to take coalition formation decisions, and in which information sharing constraints are applied, limiting this way what information agents can share and have access to, it is clear that adopting a shared network for learning would be improper. An alternative approach would be to have every agent employ its own DRL (e.g., DQN) algorithm for independent learning. However, the fact that there are episodes where some agents are inactive and thus unable to learn, poses challenges to such an approach.[2] As mentioned earlier, however, the agents could plausibly be assumed to belong to groups, or to be able to share information via social networks. Thus, instead of learning from an individual perspective, we consider the concept of learning as a social group to be a more viable alternative. Of course, this does not imply that group members have the same agent type.

One way to allow social group-level learning, would be to have a DQN employed by each group instead of by every agent. This would ensure privacy among groups, due to the fact that each group employs its own DQN network which does not communicate any information to the other groups' networks. However, employing such a solution is taxing in both resources and complexity, since the number of networks—and subsequently the number of neurons—grow larger as more agent social groups are introduced.

Instead, in our work we draw inspiration from Bootstrapped DQN [25] and similar approaches, and put forward a novel, *Privacy-Aware* Multi-Head DQN (PAMH-DQN) variant to employ in our domain. In the original Bootstrapped DQN algorithm we would have a shared network that branches into K distinct heads, where at the start of each episode a head is selected uniformly at random and used to collect experiences which can be shared across all heads [25]. A random distribution can then be used to dictate which heads may benefit from the produced experience tuples. Using this approach in our setting, however, leaves space for a social group to exploit information that may be private to its group from this experience sharing.

To avoid this, we instead propose to have a *dedicated head for each social group*, so that when an agent has to use the network, it will use the appropriate head for the group that it belongs to. In this way, an agent does not access or makes use of information that is not relevant to its group, thus not violating privacy concerns. 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

[2] Moreover, notice that simulating and evaluating such a scenario with hundreds or thousands of agents would be infeasible in practice due to computational requirements.

are acquired by its members to train upon. Additionally, there is no need for selecting a different head for each episode; every agent uses only the head assigned to its group. Furthermore, for each head we compute a different loss considering only its own target net, 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 through 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. multiplying the loss computed by each head by $1/K$. Note that the shared layers' output does not represent reward estimates, but instead encodings of the environment's states. We argue that, this approach, does not violate privacy concerns, as network outputs do not contain sensitive information, similarly to parameter sharing in [26].

4 Experimental Evaluation

In this section we present our experimental setup and the scenarios on which we test and evaluate our approach.

4.1 Experimental Setup

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 the guiding of its behavior, since it is the factor dictating the number of coalitions that the agent has the capacity to participate in the upcoming rounds. Recall that, an action consists of two elements, a coalition $\mathbf{t}_C$ and a corresponding task $\mathcal{T}_C$. The $\mathbf{t}_C$ element is a list of agent types with length between 2 and the total number of different types, though an agent type might be included more than once in the same $\mathbf{t}_C$.

A round starts with an agent being randomly chosen as the proposer, using the *softmax* distribution with temperature scaling [11, 13, 33]. The proposer then makes a proposal, which the responding agents must evaluate and respond with an acceptance or rejection decision. If a proposal is rejected, the proposer has to make a new attempt. Otherwise, the proposed coalition is formed, the agent resources allocated, the proposed task is completed, and, finally, the payoff generated is distributed equally among the agents. A round ends when the proposer has successfully reached the proposal limit p or is no longer able to make a proposal due to lack of own resources, since the proposer is allocated to all the coalitions proposed by himself, thus it is highly probable that its resources will run out in shorter time; at which point, a new proposer is chosen and another round begins. An episode, consists of many rounds, and begins with the reset of our environment, which essentially replenishes the agents' resources and resolves the formed coalitions. The end of an episode is marked when a terminal condition is met, among those discussed earlier in 3.

The resources r_i of the agents, are sampled from a uniform distribution with the lower limit being 50 and the higher 150. The same holds for the types' contribution c, which is sampled from the range between 10 and 40. Finally, our reward function and more specifically the function v draws values from a normal distribution with $\mu = 30$ and $\sigma = 20$.

We now present the parameter values that we considered in our simulations environment. First of all, we have a set of 1000 agents, where every 4 episodes, 1.5% of the total agents may be removed from, or added to the set of the available agents, simulating this way an open environment, where agents might arrive or depart in an dynamic manner. We have a total of $M = 10$ types and $L = 6$ social groups, with $n = 25$ tasks for 100 episodes, and the proposers are able to make at most $p = 10$ proposals. The threshold z_q is set to 30%. Additionally, the scenarios where we apply our *PAMH-DQN* algorithm and compare its performance against are the following two: *(i)* agents employ a shared DQN network that is trained using all the available agent experiences (*Shared-DQN*), and *(ii)* using multiple DQN networks, one for each group (*Multiple-DQN*). Also, we evaluate our networks' ability to generalize by testing already trained models in environments with modified reward functions in a transfer learning setting.

All experiments where run on a system with Intel i7-13700 CPU, with 64GB of RAM and an RTX 4060Ti-16GB GPU.

4.2 Comparison of DQN architectures

First, we examine the performance of using a *Shared-DQN* among the agents, alongside having *Multiple-DQN* and our *PAMH-DQN*, in terms of *discounted-per-episode total accumulated reward.* The discounted-per-episode total accumulated reward is calculated by applying a discount rate of δ^e to the reward $R(e)$ accumulated within each e-th episode, and progressively sum these values up to produce the $\sum_e \delta^e R(e)$ function graphs shown in the respective results figures.

This metric smooths out the reward lines and in this way eases readability. Also, it can be used to provide 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. 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. That is because an agent might not be able to form a coalition at each episode, thus the granting of rewards might be delayed. Additionally, the discount factor is reset at the beginning of each episode in our setting, thus it is difficult to obtain an overall impression regarding the evolution of the learning process. Devising a more proper metric to accurately capture sequential performance *simultaneously* within-episodes and across-episodes is left as future work.

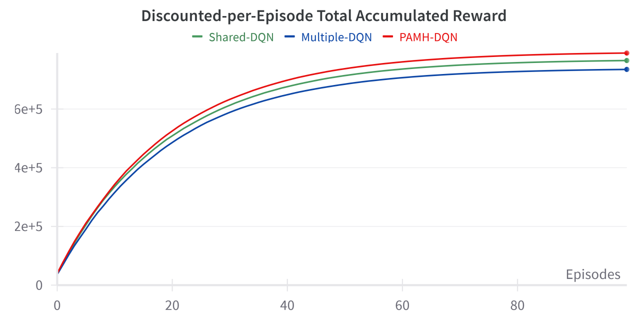

Figure 2. Discounted Accumulated Reward over 100 episodes, averaged over 10 runs. *PAMH-DQN*: $\sigma = 26,731.65$; *Shared-DQN*: $\sigma = 30,829.22$; *Multiple-DQN*: $\sigma = 102,912.25$.

Figure 2 illustrates the discounted-per-episode total accumulated reward for all the agents of our environment for 100 episodes. As we can see, our *PAMH-DQN* is the one that performs better, followed by the *Shared-DQN* network, both outperforming the *Multiple-DQN*. The numerical results are shown in Table 1. From the figures we

Table 1. Summary results for networks' performance (average of 10 runs with different seeds).

Network	Mean Discounted-per-Episode Total Accumulated Reward	Standard deviation σ	Min Reward	Max Reward
Shared-DQN	765,218.87	30,829.22	732,759.92	794,107.46
Multiple-DQN	734,955.10	102,912.25	556,375.97	806,137.22
PAMH-DQN	**790,787.27**	**26,731.65**	**750,883.91**	**825,891.82**

Table 2. Summary results for transfer learning of the evaluated approaches (average of 10 runs with different seeds).

Network	Mean Discounted-per-Episode Total Accumulated Reward	Standard deviation σ	Min Reward	Max Reward
Shared-DQN	1,510,092.40	71,220.05	1,427,975.85	1,540,210.32
Multiple-DQN	1,605,313.01	46,904.73	1,563,149.89	1,610,054.61
PAMH-DQN	**1,613,551.40**	**31,697.48**	**1,577,203.79**	**1,635,447.47**

see that *PAMH-DQN* is 7.32% better than *Multiple-DQN* and 3.29% better than the *Shared-DQN* architecture.

Our *PAMH-DQN*'s heads share some layers from the network and through the heads' training, indirectly the shared layers also get trained. As we have already mentioned (cf. Section 3) this does not violate privacy since, what is shared is trained parameters, instead of experiences or rewards, this way restraining access to sensitive information for other heads. Additionally, in the case of the shared network, the experiences of a group can be considered as noise for the groups not involved in those experiences. In *PAMH-DQN*, each head learns only from experiences involving its assigned group, nevertheless utilizing relevant common knowledge regarding the environments dynamics. Also, the fact that it achieves the lowest standard deviation illustrates its stability, in contrast to the *Multiple-DQN* approach that produces fluctuating results. In the case of the *Multiple-DQN*, although privacy is well preserved among the groups, the training process is unstable since some groups may not be as active as others in participating in coalitions.

Regarding the time performance, we report that on average, each experiment with 100 episodes lasted 480.24 seconds.

4.3 Transfer Learning Evaluation

Next, we review the performance of the aforementioned networks in an environment where we have modified the range of the distribution used in the reward function that was used in our previous experimental setup, in order to test our networks ability for transfer learning.

From the results presented in Table 2, we can see that the *PAMH-DQN* again exhibits better performance than the rest of the methods. It surpasses *Multiple-DQN*, even if not by a large margin, in terms of mean discounted-per-episode total accumulated reward, and also is more stable, judging by the standard deviation. Interestingly, in these trials also, *PAMH-DQN* managed to achieve the highest reward among the best achieved by all methods, as indicated in the rightmost column of the table. Once again, for the *Shared-DQN* it appears that the learning of the groups is affected by the fact that the network is trained on all experiences regardless of each group's involvement.

The ability of *PAMH-DQN* (and secondarily of *Multiple-DQN*) to learn a policy for each group of agents, allows for more robust performance and better generalization. In addition, the fact that each social group is trained on the data that it collects, leads to better adaptability in complex environments. This feat also hints on *PAMH-DQN* being a good fit for non-stationary environments.

5 Conclusions and Future Work

In this work, we have developed a training framework about task execution in multiagent scenarios, where agents belonging in social groups form coalitions and learn the value of their cooperative activities in a complex, open, environment. To this end, we have investigated the use of different DRL methods to tackle this demanding problem. These methods include the use of a shared-network architecture, of multiple distinct DQN networks, and of our novel algorithm that utilizes a multi-head model. Via experimental evaluation, we showed that our PAMH-DQN strikes a balance between complexity and performance, all while preserving privacy between distinct agent groups.

In ongoing and future work, we intend to experiment with alternative (possibly normalized) reward functions to further test and increase the stability of our algorithm, as well as design a metric to more accurately reflect the sequential performance of methods, specifically for this setting. Additionally, we want to incorporate type uncertainty to our environment, which will make our learning more challenging but also allow for an increased transfer learning potential. Such an approach can also be paired with the incorporation of Graph Neural Networks (GNNs) [37] in our setting. GNNs bear the potential to effectively capture the structure of an environment involving multiagent interactions, and thus more accurately represent the agents' relationships within their respective coalitions or groups as well as among their types.

References

[1] Y. Bachrach, R. Everett, E. Hughes, A. Lazaridou, J. Z. Leibo, M. Lanctot, M. Johanson, W. M. Czarnecki, and T. Graepel. Negotiating team formation using deep reinforcement learning. *Artificial Intelligence*, 288, 2020.

[2] G. Chalkiadakis and C. Boutilier. Bayesian reinforcement learning for coalition formation under uncertainty. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 1090–1097, 2004.

[3] G. Chalkiadakis and C. Boutilier. Sequential decision making in repeated coalition formation under uncertainty. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, page 347–354, Richland, SC, 2008. ISBN 9780981738109.

[4] G. Chalkiadakis and C. Boutilier. Sequentially optimal repeated coalition formation under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 24:441–484, 2012.

[5] G. Chalkiadakis, E. Markakis, and C. Boutilier. Coalition formation under uncertainty: Bargaining equilibria and the bayesian core stability concept. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007.

[6] G. Chalkiadakis, E. Elkind, E. Markakis, M. Polukarov, and N. R. Jennings. Cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, 39:179–216, 2010.

[7] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory (Synthesis Lectures on Artificial Inetlligence and Machine Learning)*. Morgan & Claypool Publishers, 2011.

[8] R. Chen, C. Yi, K. Zhu, J. Cai, and B. Chen. A DRL-based hierarchical game for physical layer security with dynamic trilateral coalitions. In *ICC 2023 - IEEE International Conference on Communications*, pages 4495–4500, 2023. doi: 10.1109/ICC45041.2023.10279584.

[9] E. Elkind, G. Chalkiadakis, and N. R. Jennings. Coalition structures in weighted voting games. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, page 393–397, NLD, 2008. IOS Press. ISBN 9781586038915.

[10] V. Franois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and JoellePineau. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018. doi: 10.1561/2200000071. URL https://doi.org/10.48550/arXiv.1811.12560.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. volume 70 of *Proceedings of Machine Learning Research*, 08 2017.

[12] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33:750 – 797, 2018. URL https://api.semanticscholar.org/CorpusID:202540003.

[13] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*, 2015.

[14] S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, pages 193–202, 2005.

[15] A. A. Khan and R. Adve. Centralized & distributed deep reinforcement learning methods for downlink sum-rate optimization. *CoRR*, abs/2009.03033, 2020. URL https://arxiv.org/abs/2009.03033.

[16] S. Kraus, O. Shehory, and G. Taase. The advantages of compromising in coalition formation with incomplete information. pages 588– 595, 02 2004. ISBN 1-58113-864-4. doi: 10.1109/AAMAS.2004.242428.

[17] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *CoRR*, abs/1511.06314, 2015. URL http://arxiv.org/abs/1511.06314.

[18] Y. Liu, H. Zhou, Y. Deng, and A. Nallanathan. Channel access optimization in unlicensed spectrum for downlink urllc: Centralized and Federated DRL approaches. *IEEE Journal on Selected Areas in Communications*, 41(7):2208–2222, 2023. doi: 10.1109/JSAC.2023.3280982.

[19] H. A. Mahdiraji, E. Razghandi, and A. Hatami-Marbini. Overlapping coalition formation in game theory: A state-of-the-art review. *Expert Systems with Applications*, 174:114752, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.114752. URL https://www.sciencedirect.com/science/article/pii/S0957417421001937.

[20] M. Mamakos and G. Chalkiadakis. Overlapping coalition formation via Probabilistic Topic Modeling. AAMAS '18, page 2010–2012, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop 2013*, 2013.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

[23] M. Moafi, R. R. Ardeshiri, M. W. Mudiyanselage, M. Marzband, A. Abusorrah, M. Rawa, and J. M. Guerrero. Optimal coalition formation and maximum profit allocation for distributed energy resources in smart grids based on cooperative game theory. *International Journal of Electrical Power & Energy Systems*, 144:108492, 2023. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2022.108492.

[24] S. Orfanoudakis and G. Chalkiadakis. A novel aggregation framework for the efficient integration of distributed energy resources in the smart grid. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 06 2023.

[25] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via Bootstrapped DQN. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf.

[26] J. Qi, Q. Zhou, L. Lei, and K. Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *ArXiv*, abs/2108.11887, 2021. URL https://api.semanticscholar.org/CorpusID:237303858.

[27] N. Qi, Z. Huang, F. Zhou, Q. Shi, Q. Wu, and M. Xiao. A task-driven sequential overlapping coalition formation game for resource allocation in heterogeneous UAV networks. *IEEE Transactions on Mobile Computing*, 22(8):4439–4455, 2023. doi: 10.1109/TMC.2022.3165965.

[28] M. Sadeghi and M. Erol-Kantarci. Deep reinforcement learning based coalition formation for energy trading in smart grid. In *2021 IEEE 4th 5G World Forum (5GWF)*, pages 200–205. IEEE, 2021.

[29] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1):165–200, 1998. ISSN 0004-3702. URL https://www.sciencedirect.com/science/article/pii/S0004370298000459.

[30] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[31] A. Taghizadeh, M. Montazeri, and H. Kebriaei. Deep reinforcement learning-aided bidding strategies for transactive energy market. *IEEE Systems Journal*, 16(3):4445–4453, 2022. doi: 10.1109/JSYST.2022.3145102.

[32] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. AAAI'16, page 2094–2100, 2016.

[33] P.-H. Wang, S.-I. Hsieh, S.-C. Chang, Y.-T. Chen, J.-Y. Pan, W. Wei, and D.-C. Juan. Contextual temperature for language modeling. *arXiv preprint arXiv:2012.13575*, 2020.

[34] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3): 279–292, 05 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

[35] T. Wolff and A. Nieße. Dynamic overlapping coalition formation in electricity markets: An extended formal model. *Energies*, 16(17):6289, 2023.

[36] M. Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[37] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

[38] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng. Privacy-preserving federated deep learning with irregular users. *IEEE Transactions on Dependable and Secure Computing*, 19(2):1364–1381, 2022. doi: 10.1109/TDSC.2020.3005909.

[39] 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 Transactions on Parallel and Distributed Systems*, 24(5): 1042–1051, 2013. doi: 10.1109/TPDS.2012.213.

[40] M. Yokoo, V. Conitzer, T. Sandholm, N. Ohta, and A. Iwasaki. A new solution concept for coalitional games in open anonymous environments. pages 53–64, 2006. doi: 10.1007/11780496_6. Joint JSAI 2005 Workshop on New Frontiers in Artificial Intelligence ; Conference date: 13-06-2005 Through 14-06-2005.

[41] J. Zhang, Z. Yu, S. Mao, S. C. Periaswamy, J. Patton, and X. Xia. Iadrl: Imitation augmented deep reinforcement learning enabled ugv-uav coalition for tasking in complex environments. *IEEE Access*, 8: 102335–102347, 2020.