Technical University of Crete

School of Electrical and Computer Engineering

Deep Reinforcement Learning for Overlapping Coalition Formation

Gerasimos Koresis

Thesis Commitee

Supervisor: Professor Georgios Chalkiadakis (ECE)

Professor Michail G. Lagoudakis (ECE)

Professor Vasileios Samoladas (ECE)

Chania 2023

Abstract

This thesis delves into the dynamic landscape of Overlapping Coalition Formation (OCF), within multiagent systems, leveraging the power of Deep Reinforcement Learning (DRL), to navigate uncertainties inherent in cooperative interactions.

The central inquiry revolves around addressing the uncertainty regarding the degrees of cooperation (DoC) among agent *types*, which arguably determine the effectiveness of coalitions formed by the agents.

The study unfolds in multiple dimensions. First, an exploration of RL and DRL techniques is undertaken, emphasizing their application to the intricate challenges posed by OCF scenarios. The core of the investigation lies in deciphering the evolving dynamics of agent interactions, with a particular focus on the uncertain nature of cooperation values represented by the DoC.

In response to this uncertainty, the study integrates Graph Neural Networks (GNN) into the DRL framework. In particular, our thesis details the synergistic integration of DRL (specifically, Deep Q-Networks - DQN) and GNNs (specifically, Graph Attention Networks - GAT), showcasing their collective capacity to adapt to the ever-changing uncertain cooperation landscape. Our experimental evaluation results underscore the efficacy of this hybrid approach in enhancing sequential coalition formation strategies under uncertainty.

We explored several variants of our DRL+GNNs approach, with our simulation results suggesting the intertwining of DQN with GAT updates of the DoC at the change of the proposer to be the most beneficial one.

Finally, our work in this thesis takes the initial steps to tackle the scalability challenges inherent in this multiagent domain, and lays the groundwork for future refinements and extensions.

Abstract

Η διπλωματική αυτή, εμβαθύνει στο δυναμικό τοπίο του Σχηματισμού Επικαλυπτώμενων Συνασπισμών (*Overlapping Coalition Formation-OCF*), σε πολυπρακτορικά συστήματα, αξιοποιώντας την ισχύ της Βαθιάς Ενισχυτικής Μάθησης (*Deep Reinforcement Learning-DRL*), για να πλοηγηθούμε στις αβεβαιότητες που είναι εγγενείς στις συνεργατικές αλληλεπιδράσεις.

Η έρευνα περιστρέφεται γύρω από την αντιμετώπιση της αβεβαιότητας σχετικά με τους βαθμούς συνέργειας (*Degrees of Cooperation-DoC*) ανάμεσα στους τύπους πρακτόρων, που συζητήσιμα καθορίζει την αποτελεσμάτικοτητα των συνασπισμών που σχηματίζουν οι πράκτορες.

Η μελέτη εκτυλίσσεται σε πολλαπλές διαστάσεις. Πρώτον, πραγματοποιείται μια εξερεύνηση των τεχνικών RL και DRL, δίνοντας έμφαση στην εφαρμογή τους στις περίπλοκες προκλήσεις που θέτουν τα σενάρια OCF. Ο πυρήνας της έρευνας έγκειται στην αποκρυπτογράφηση της εξελισσόμενης δυναμικής των αλληλεπριδράσεων των πρακτόρων, με ιδιαίτερη έμφαση στην αβέβαιη φύση των τιμών συνεργασίας που αντιπροσωπεύονται απο τα DoC.

Ως απάντηση σε αυτή την αβεβαιότητα, η μελέτη μας ενσωματώνει τα νευρωνικά δίκτυα σε γράφους (*Graph Neural Networks-GNN*) στο πλαίσιο του DRL. Ειδικότερα, η εργασία μας περιγράφει λεπτομερώς τη συνεργιστική ολοκλήρωση των DRL (συγκεκριμένα, $DeepQ-Networks$ - DQN) και GNN (συγκεκριμένα, *Graph Attention Networks* - GAT), επιδεικνύοντας την ικανότητα του συνδυασμού τους να προσαρμόζεται στο διαρκώς μεταβαλλόμενο αβέβαιο τοπίο συνεργασίας. Τα αποτελέσματα της πειραματικής μας αξιολόγησης υπογραμμίζουν την αποτελεσματικότητα αυτής της υβριδικής προσέγγισης στη βελτίωση των ακολουθιακών αποφάσεων σχηματισμού συνασπισμού υπό αβεβαιότητα.

Εξερευνήσαμε διάφορες παραλλαγές της DRL+GNN προσέγγισής μας, με τα αποτελέσματα της προσομοίωσής μας να υποδηλώνουν ως την πλέον επωφελή την αλληλοδιαπλοκή του DQN με ενημερώσεις GAT των DoC που λαμβάνουν χώρα κάθε φορά που αλλάζει ο προτείνων τον σχηματισμό.

Τέλος, στην εργασία μας πραγματοποιούμε κάποια αρχικά βήματα για την αντιμετώπιση των εγγενών σε αυτό το πολυπρακτορικό πρόβλημα προκλήσεων επεκτασιμότητας, και θέτει τις βάσεις για μελλοντικές βελτιώσεις.

Acknowledgements

First and foremost, I would like to express my deepest gratitude towards my supervisor, Professor Georgios Chalkiadakis, for his guidance and trust in the journey of this thesis, and for greatly motivating me to further expand my studies on topics related to Artificial Intelligence. I would also like to thank the members of the Intelligence Systems Lab, Dimitris, Errikos and Iasonas, for our discussions and their helpful suggestions.

Second, I wish to thank the members of the commitee, Professor Michail G. Lagoudakis and Professor Vasileios Samoladas, for their helpful comments and time.

Last, but not least, I deeply thank my dear family and friends for all their unconditional love and support, not only during the course of my studies, but in every part of my life.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Deep Reinforcement Learning or DRL for short, is a powerful branch of machine learning that combines Reinforcement Learning (RL) algorithms with deep neural networks. Unlike traditional reinforcement learning, which is well-suited for single-agent environments, DRL allows multiple agents to interact with their environment simultaneously and learn optimal strategies through trial and error. The combination of reinforcement learning with deep neural networks, made it possible to tackle problems that up until that point were unfeasible for a machine. It has gained significant attention due to its ability to handle high-dimensional state and action spaces, making it well-suited for various applications such as game playing, namely it has successfully learn to play ATARI games [25], robotics and autonomous decision-making [47].

In the context of coalition formation, DRL offers several advantages. First, DRL enables agents to make decisions based on incomplete and dynamic information, which is common in real-world scenarios[51]. Agents can learn to adapt their coalition formation strategies, leading to more flexible and adaptive coalitions. Second, DRL can handle large and continuous action spaces[34][38], making it well-suited for scenarios with numerous potential coalitions and cooperation options[61]. The ability to represent complex cooperation strategies through deep neural networks empowers agents to learn sophisticated and context-dependent cooperation behaviors [39].

By leveraging the power of DRL, coalition formation problems can be solved more efficiently and effectively[62]. Agents can learn to form stable and high-performing

coalitions, leading to improved resource allocation, task allocation, and overall system efficiency.

Overlapping coalition formation, on the other hand, refers to the process of creating groups or coalitions in which individuals can participate in multiple coalitions simultaneously [13]. Coalition formation is a fundamental concept in multi-agent systems and has been extensively studied in various domains, including cooperative robotics [11], social network analysis [29], and resource allocation [18]. The emergence of overlapping coalitions introduces additional complexity to the coalition formation problem, as agents need to strategically decide which coalitions to join or leave, taking into account the potential benefits and conflicts among different coalitions [20]. The extension of coalition formation, that is overlapping coalition formation, brings various benefits to a multi-agent system.

One of the major advantages of overlapping coalition formation is the improved resource utilization, since agents can allocate their resources more efficiently by taking part in multiple coalitions that complete tasks to gather reward. That aspect enables agents to exploit their diverse capabilities effectively, leading to a higher task completion rate and better overall system performance [54] [13] [53].

Overlapping coalition formation also fosters increased collaboration and knowledge sharing among agents. Agents are able to learn from their interactions in different coalitions and transfer knowledge and experience across multiple tasks [8]. This knowledge exchange facilitates collective learning and enables agents to make more informed decisions during coalition formation. Moreover, the dynamic nature of overlapping coalitions allows agents to adapt to changing environments.

Now, sequential coalition formation is a natural coaltion formation paradigm that handles cases where agents form coalitions in order to complete tasks in a sequential manner over time, presenting a significantly different approach from the traditional simultaneous coalition formation settings [7]. The sequential nature of coalition formation introduces new complexities and challenges. Agents must make decisions while taking into account incomplete information about the co-operational

values between the types of agents [12]. The need for adaptive and forward-looking strategies becomes crucial to forming effective coalitions [19].

In sequential coalition formation, agents have to face trade-offs between immediate and future rewards. A well-designed strategy should be able to balance the benefits of joining the current coalition against the possibility of forming a more rewarding one in the future [57]. In order to address such challenges, dynamic optimizations techniques and online decision-making algorithms are often employed in sequential coalition formation. Reinforcement learning methods, including deep reinforcement learning, can offer promising avenues for the agents to learn effective coalition formation strategies in a sequential environment [14]. By making decisions in an iterative manner, they can continually update their cooperation values and learn from past experiences to form more successful and efficient coalitions over time.

The combination of overlapping coalition formation and sequential coalition formation presents a fascinating research direction with promising applications. Moreover, in the deep learning era, it is worth examining whether deep reinforcement learning (DRL) techniques are able to provide a robust framework for tackling there complexities and learning effective coalition formation strategies in dynamic, real-world multi-agent systems [63]. By leveraging the capabilities of DRL, one could expect to develop intelligent agents that can learn to collaborate with multiple groups simultaneously, adapt to changing circumstances, and make strategic decisions based on a combination of individual and collective objectives.

In our case, we want to explore the combination of the aforementioned strategies and techniques, in an environment where agents that belong to agent types, form coalitions in order to complete tasks and gather rewards sequentially, which in a way means by also forming overlapping coalitions. The agents in our setting face the fact that they do not know the true co-operational values between the types and thus have to act using estimates of such values.

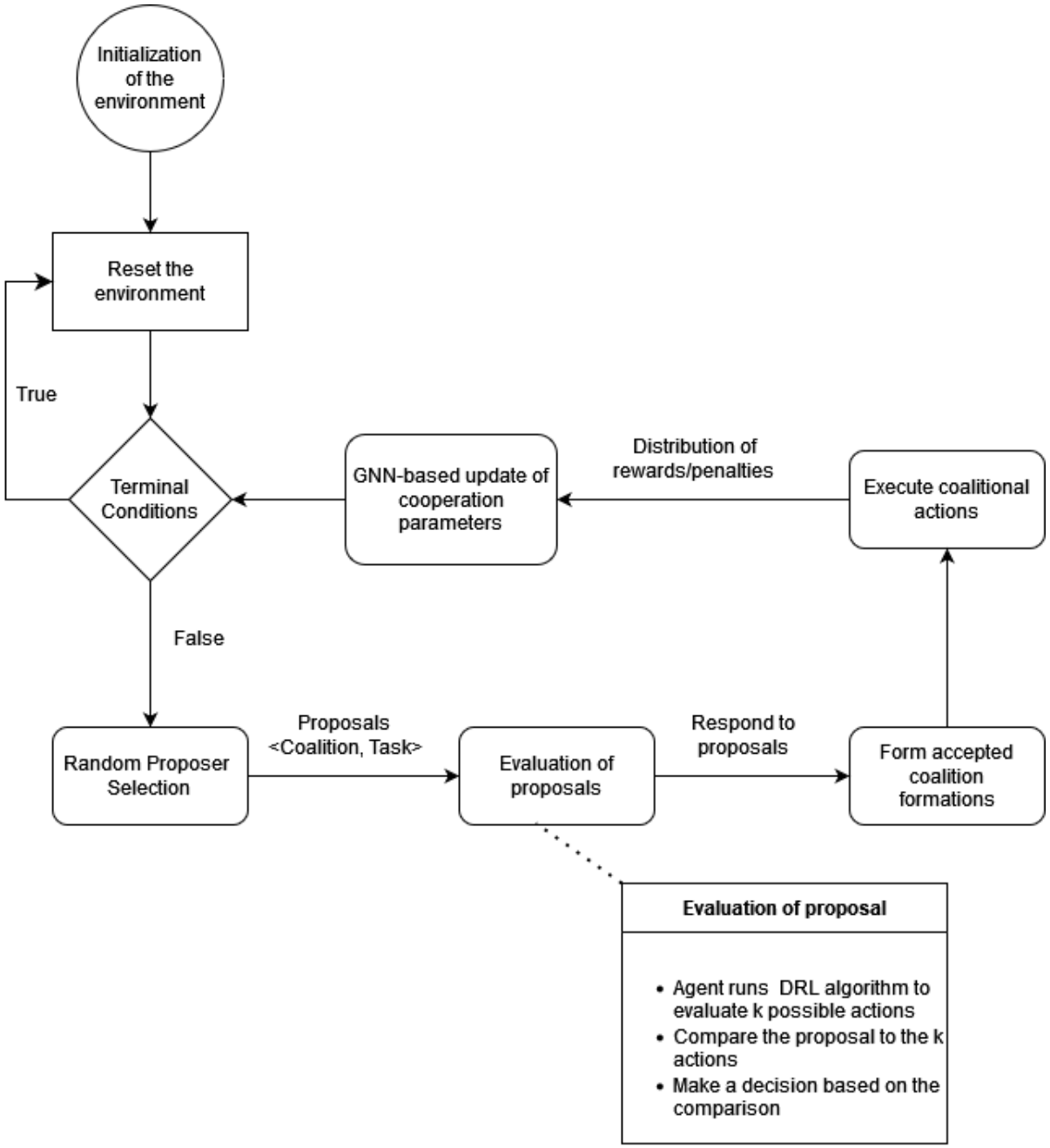

Figure 1.1: A DRL sequential coalition formation framework with values of agent type-related uncertainty captured via unknown cooperation parameters (Degrees of Cooperation)

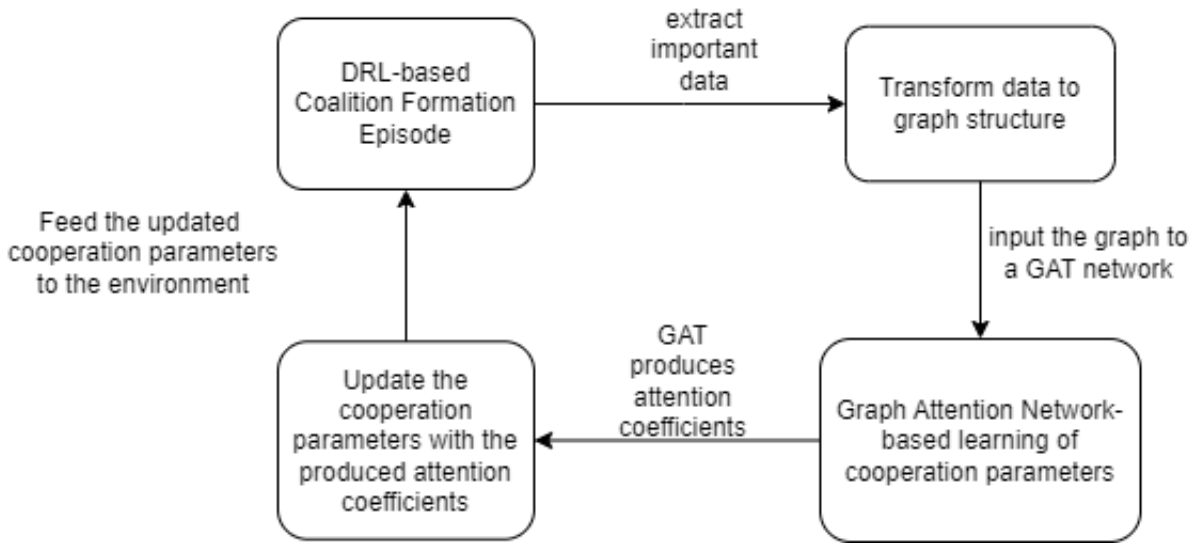

Figure 1.2: A Graph Attention Network learning of the cooperation parameters

Henceforth we will refer to our formation model with the term DRL-based coalition formation, where an agent is chosen at each round to be the proposer, making proposals to other agents, that consist of both a coalition to be formed and additional action to perform (in order to fulfill a task). The coalitions are formed sequentially and thus can be overlapping. Then the accepted coalitions are formed and the actions that they agreed to are performed. According to the outcome of the action, rewards and penalties are equally distributed equally among the agents that participated. Afterwards, we update the cooperation values between the agent types, either by using Graph Neural Networks-based learning (GAT [37]) or by a simpler update protocol that we put forward. Lastly, the updated cooperation values are provided to the agents to start again the coalition formation process, this process continues until a terminal condition is met (will be explained in more detail in the appropriate section), for example if no agent with enough resources to participate in a coalition remains.

1.1 Contributions

In this thesis, we aim to harness the potential of using DRL techniques in the domain of overlapping coalition formation [23]. Through our investigation, we seek to adapt and apply cutting-edge DRL algorithms to efficiently form and manage over-

lapping coalitions in a dynamic, sequential manner. By analyzing the performance, scalability, and robustness of these novel approaches, we aim to unlock valuable insights into the benefits and limitations of using DRL in the context of overlapping coalition formation.

Through the combination of the powerful strengths of DRL [33] with the innovative concept of overlapping coalition formation [13] as well as leveraging the dynamics of sequential coalition formation [19], we anticipate novel contributions that can advance our understanding of collective decision-making, coordination, and cooperation in multi-agent systems. Our approach opens the door to fresh perspectives on how autonomous agents can effectively collaborate, adapt, and optimize their resource utilization in complex and ever-changing environments.

In the episodic setting of our research, we adopt a carefully designed protocol where an agent, randomly selected as the proposer at each turn, sequentially makes coalition formation proposals. This unique design enables the formation of overlapping coalitions, paving the way for enhanced resource allocation and flexible task distribution among agents.

One main challenge we tackle in our work is to allow learning in the face of uncertainty associated with cooperation values between different types of agents, we address this challenge in an innovative manner, by employing the power of Graph Neural Networks. By doing so, our work seeks to achieve a major breakthrough in enhancing the agents' trust and predictive accuracy when forming coalitions. We believe that this improvement will not only lead to more efficient and successful coalitions, but will also set the stage for transformative advancements in various domains where multi-agent cooperation is essential.

To our knowledge, this is the first time such a model-free approach with some model-based elements is implemented with the exception of [35], which follows a similar path. We say some model-based elements, due to the fact that agents keep information from their experiences trying to find the cooperation values between agent types.

In conclusion, our thesis embarks on an exciting journey to unlock the untapped potential of DRL in the realm of overlapping coalition formation. We invite readers to delve into our findings, explore the intricacies of our methodology, and join us in trying to build an environment where agents collaborate seamlessly and dynamically to conquer complex challenges.

1.2 Outline

In Chapter 2 we will review some of the existing literature on the foundational concepts and algorithms of DRL as well as examine successful applications of DRL in various domains, emphasizing its potential in multi-agent systems. We will also look into the literature on traditional coalition formation approaches and challenges.

Our focus though will be on the specific topic of overlapping coalition formation [13], discussing its advantages and relevance in dynamic environments where agents face uncertainty [10], [12], [51].

Furthermore, we will explore the synergy between overlapping coalition formation [54] and DRL techniques [62], as well as analyze how DRL algorithms could be adapted to address the complexities of forming and managing overlapping coalitions.

Moving on, we will cover the theoretical foundations of DRL [25] by providing an in-depth explanation of essential DRL concepts, including Markov Decision Processes (MDPs) [2], [21] and Q-learning [5], [41]. Additionally, we will present key DRL algorithms such as Deep Q-Networks(DQNs)[25], [41] and Double DQNs [28], [43].

After that, we delve into the theoretical aspects of sequential overlapping coalition formation, defining and characterizing them and their advantages in multi-agent systems. As well as discuss the challenges and potential solutions for forming and managing overlapping coalitions.

Next, in Chapter 3 we present you with related work and research that helped us in the formulation and realization of our approach, giving you an insight in other similar or noteworthy papers in the field of DRL, Overlapping and Sequential

Coalition Formation as well as in Graph Neural Networks (GNNs).

Then in Chapter 4 comes the presentation of our methodology where we define the problem formulation and the specific objectives of our research and describe its key concepts. As well as detail the DRL-based approach on forming and managing overlapping coalitions in a sequential manner and the experimental setup used to evaluate our DRL-based approach.

In Chapter 5, we make a performance evaluation of the DRL Overlapping Coalition Formation showing the results of our experiments and simulations conducted to assess our DRL-based approach. We also discuss the advantages and limitations of our approach over existing techniques in order to make a substantial comparison.

Lastly, we end our presentation by summarizing our thesis contributions, then discuss the broader implications of our research findings for the field of combining DRL with overlapping coalition formation, which occurs in a sequential manner.

We conclude our thesis with closing remarks regarding our findings and suggest directions for further research and development in that area.

Chapter 2

Theoretical Background

In this chapter, we present the theoretical foundations of our research on utilizing Deep Reinforcement Learning (DRL) techniques for overlapping coalition formation. With the increasing complexity and the dynamic nature of multi-agent systems, traditional coalition formation approaches often face difficulties in managing to allocate resources and distribute tasks efficiently.

To battle against these challenges, we turn to DRL - a powerful paradigm that enables agents to learn effective decision-making strategies by interacting with the environment.

As such, this chapter lays the groundwork for the thesis, by providing a comprehensive review of the key concepts of DRL, as well as the theoretical aspects of overlapping coalition formation and sequential coalition formation and the basics for Graph Neural Networks, that are going to be used later on.

2.1 Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL)

In this section we will introduce the fundamental concepts of Deep Reinforcement Learning and algorithms used in that field. We will start with the Markov Decision Processes (MDPs) a formal framework for modeling sequential decision-making problems in uncertain environments. Subsequently, we delve into Q-learning[41], a

popular off-policy algorithm used in reinforcement learning and then introduce the Deep Q-Networks (DQNs) as an extension of Q-learning that uses neural networks to approximate the Q-function[25]. Additionally, we explore policy gradient methods, an alternative approach to DRL that directly optimizes the policy function.

The insights gained from the comprehensive exploration that was stated above, we lay the groundwork for the adaptation and expansion of DRL algorithms to address the challenges of overlapping coalition formation.

2.1.1 Markov Decision Processes (MDPs)

The Markov decision process (MDP) is a mathematical framework used for modeling decision-making problems where the outcomes are partly random and partly controlled by a decision maker, which makes sequential decisions over time. MDPs evaluate which actions should be taken through evaluation of the current state and environment of the system. They provide a formal structure for modeling a wide range of real-world problems, including robotics [32], game playing[21], finance [17], and most importantly, multi-agent systems[55], [52].

The MDP model operates by using key elements such as the agent, states, actions, rewards, and optimal policies. The agent refers to a system responsible for making decisions and performing actions. It operates in an environment that details the various states that the agent is in, providing the information he needs to make decisions. Actions correspond to the choices an agent can make to transition from one state to another. MDP defines the mechanism of how certain states and an agent's actions lead to the other states, influenced by transition probabilities. Moreover, the agent receives immediate rewards or penalties depending on the action it performs and the state it attains (current state). These rewards guide the agent's learning process by indicating the desirability of its actions. The policy for the MDP model reveals the agent's following action depending on its current state [2].

The graphical representation of an MDP model is the following:

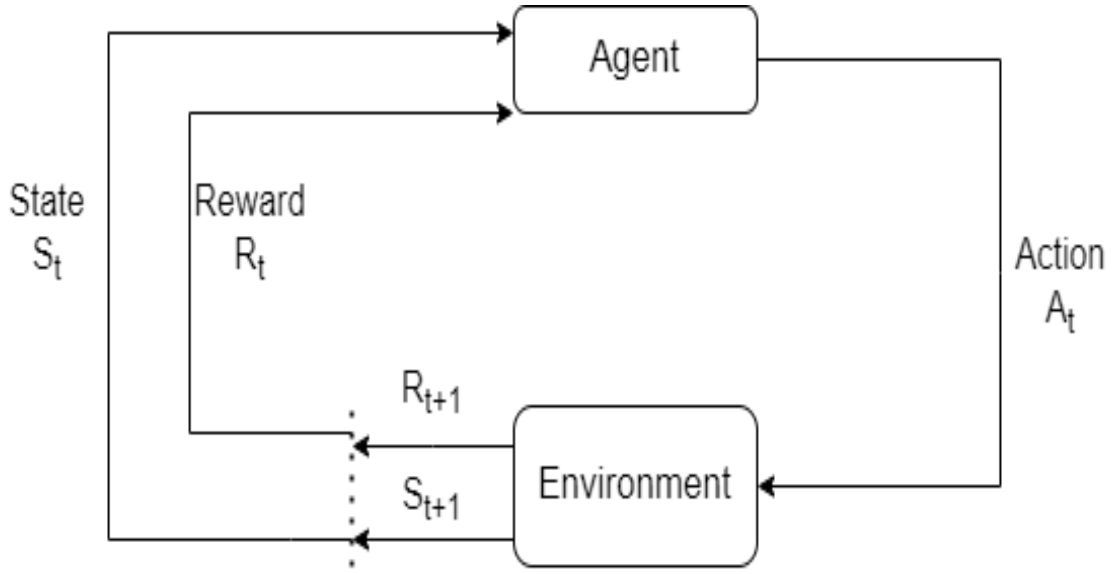

Figure 2.1: MDP model

The MDP model makes use of the Markov Property, which states that the future can be determined only from the present state that encapsulates all the necessary information from the past. The Markov Property can be evaluated by using this equation:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, S_3, \dots S_t] \tag{2.1}$$

What this equations says, is that the probability of the next state ($\mathbf{P}[S_{t+1}]$) given the present state (S_t) is given by the next state's probability considering all previous states ($S_1, S_2, S_3, \dots S_t$).This implies that MDP uses only the current state to evaluate the next actions without any dependencies on previous states or actions.

Another crucial component of the MDP framework is that it allows agents to adopt policies, which essentially dictate the agent's behaviour through associating actions with states. Policy iteration and value iteration [2] are algorithms used to compute the optimal policy and value function, respectively. Policy iteration alternates between policy evaluation and policy improvement steps until convergence. On the other hand, value iteration updates the value function directly in an iterative manner, converging to the optimal values and policy. A policy (π) is considered optimal when it is able to determine the agent's optimal action given the current

state, in order to maximize the expected cumulative reward over time. To that end, a variable termed 'discount factor (γ)' is introduced. The closer it is to zero, the immediate rewards get prioritized, while when it is closer to one, the focus shifts to long-term rewards [21]. Therefore it plays a key role to finding the best policy for our agent. Its formula is characterized by the expected sum of discounted future rewards:

$$\mathbf{V}(s) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_t] \tag{2.2}$$

The value function comprises of two components: the reward of the current state and the discounted reward value of the next state. It is derived from a fundamental equation, which we will cover next.

At the core of MDPs lies the Bellman equation, a recursive equation that expresses the optimal value of a state as the maximum expected cumulative reward achievable from that state. From that equation is (2.2) derived from.The optimal value function, often denoted as $V^*(s)$, quantifies the expected sum of discounted rewards that an agent can accumulate by following an optimal policy. The Bellman equation forms the cornerstone of dynamic programming approaches, enabling the calculation of optimal values and policies. The Bellman equation for the optimal value function is shown below:

$$\mathbf{V}^*(s) = \max_a [R(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a) V^*(s')] \tag{2.3}$$

Although MDPs provide a powerful framework for modeling decision-making, it also comes with challenges, especially in complex and high-dimensional environments. In the case of a dynamic state and action space, the computational complexity of solving MDPs is increased exponentially, presenting us with scalability issues. Moreover, to assume having fully observable states and known transition probabilities may not be applicable to real-world scenarios, such as dealing with uncertainty, like in our setting, which requires sophisticated modeling techniques.

To that end, MDPs have been extended and modified to accommodate var-

ious real-world considerations. Partially Observable Markov Decision Processes (POMDPs)[60] address scenarios where the agent has limited observability of the environment[33]. Continuous state and action spaces are handled by Continuous MDPs (CMDPs) or by discretizing the spaces [58], [45]. Multi-agent environments introduce Multi-Agent Markov Decision Processes (MMDPs), which involve multiple agents interacting in a shared environment, each with their policies and rewards[52].

In the end, Markov Decision Processes, serve as the foundational framework for understanding sequential decision-making under uncertainty. The integration of MDPs with reinforcement learning techniques has formed the basis for various algorithms, including Q-learning [5], which we will cover in the next section, and policy gradient methods, who play a pivotal role to the exploration of deep reinforcement learning for sequential overlapping coalition formation.

2.1.2 Q-learning and Deep Q-Networks(DQNs)

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. It is a powerful paradigm used in decision-making in sequential environments.

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state [5]. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations. The main focus of Q-learning is to make accurate estimation of the Q-function, which is used to represent the expected cumulative reward an agent is able to obtain through taking specific actions in given states.

The Q-function helps the agent to select which action to take, enabling him to overtime learn how to maximize the rewards so as to behave optimally at any state he is in.

A Q-learning model, works in an iterative process where there are several core components working together in order to facilitate the learning process. These

include the state representation, where the environment is represented by states, the action selection, where the agent chooses action based on the Q-values, and the reward function, which provides valuable feedback on the agent's actions. Crucial part in the above elements coming together plays the Bellman equation which forms the foundation of Q-learning, expressing the optimal Q-values in terms of immediate rewards and the maximum Q-value of the next state.

The main principle of Q-learning lies in the Q-value update, which iteratively refines the Q-values during the learning process. At each time step, the agent takes an action, receives a reward from it and transitions to a new state. The Q-value update is based on the Bellman equation, which allows the agent to update the Q-value for the current state-action pair by using a combination of the observed reward and the maximum Q-value of the next state[36]. This process iterates until convergence to approximate the optimal Q-values. The Bellman equation has the following form:

$$\underbrace{\text{New}Q(s,a)}_{\text{New Q-Value}} = Q(s,a) + \underbrace{\alpha}_{\text{learning rate}}[\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}} \overbrace{\max Q'(s',a')}^{\text{Maximum predicted reward, given new state and all possible actions}} - Q(s,a)] \tag{2.4}$$

The equation breaks down in the following terms. The $Q(s,a)$ represents the expected reward for taking action a in state s. The actual reward received by that action is referenced by R while s' refers to the next state. The learning rate is α and the discount factor is γ. The highest expected reward for all possible actions a' in state s' is represented by $max(Q(s',a'))$.

While Q-learning is a fundamental reinforcement learning algorithm, it faces certain challenges and limitations, particularly in dynamic and high-dimensional environments. The main challenge lies in the need for large state-action spaces, which may render traditional Q-learning computationally infeasible. 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[36].

Algorithm 1 Q-Learning Algorithm [46]

Input: Q-table with random values or zeros
Output: Learned Q-values for state-action pairs
for each episode **do**
  Initialize environment, get initial state s
  while episode not finished **do**
    Choose action a using exploration strategy (e.g., epsilon-greedy)
    Take action a, observe reward r and next state s'
    Update Q-value for state-action pair (s, a)
    $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)]$
    Move to the next state $s' \leftarrow s$
    if s' is terminal **then**
      Break the loop

To address the limitation of traditional Q-learning, Deep Q-Networks (DQNs) were introduced as a groundbreaking extension [28]. DQNs take advantage of the power of neural networks to approximate the Q-function, enabling the handling of complex and high-dimensional state-action spaces. By using deep neural networks, DQNs are able to learn intricate representations and capture the underlying structures of the environment[25]. DQNs consist of multiple layers, including input layers that receive the state information, hidden layers that process the data, and output layers that produce the Q-values for each action. Convolutional Neural Networks (CNNs) are commonly used as the hidden layers, particularly for environments with visual inputs. CNNs excel at learning spatial features, making them well-suited for tasks like playing Atari games[22].

One of the major innovations that DQN brought was the use of experience replay. Instead of updating the Q-values after every experience, DQN stores experiences in a buffer and samples it randomly during training. Experience replay enhances the stability of the learning process and promotes better data efficiency, as experiences are reused multiple times to break the temporal correlations between consecutive experiences. DQN also employs a target network to address challenges related to target value estimation during training. The target network is a copy of the DQN used for predicting the Q-values, but its parameters are only periodically updated (after a number of iterations of the main network) to provide more stable and consistent target values for Q-values updates. By transferring the weights of the

main network to the target network, we manage to transfer the knowledge learned from one to the other. This mechanism mitigates the problem of moving target values during training and prevents undesirable tribulations [33]. The above results in a modified Bellman equation for DQN as shown below:

$$Q(s, a; \theta) = r + \gamma \max Q(s', a'; \theta') \tag{2.5}$$

As we can see, the Q-functions are parameterized by the network weights θ and θ'. The weight vector θ is used to represent the main neural network which is used to estimate the Q-values for the current state s and action a, while the functionality of the weight vector θ' is to parameterize the target neural network, that estimates the Q-values for the next state s' and action a'.

The training process of a DQN involves selecting an appropriate loss function, typically the mean square error, to quantify the discrepancy between the predicted Q-values and target Q-values. The loss function for the training neural network has the following form:

$$L(\theta) = \mathbb{E}[(r + \gamma \max Q(s', a'; \theta') - Q(s, a; \theta))^2] \tag{2.6}$$

As we can see, the loss for the DQN algorithm is defined as the mean square error between the two sides of the Bellman equation (2.5). Stochastic Gradient Descent (SGD) or its variants are used for optimization, updating the neural networks parameters in an iterative manner in order to minimize the loss function[33]. Hyperparameters, such as the learning rate and the batch size, play a crucial part in the training process and need to be carefully tuned so as to achieve optimal performance.

Algorithm 2 Deep Q-learning with experience replay [25]

Initialize replay memory D to capacity N,
Initialize action-value function Q with random weights θ,
Initialize target action-value function $\hat{Q}$ with random weights $\theta^- = \theta$
for episode = 1, M **do** do
  Initialize sequence $s_1 = x_1$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for t = 1, T **do** do
    With probability ϵ choose a random action a_t
    otherwise select $a_t = \arg\max_\alpha(Q(\phi(s_t)), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and process $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

$$Set\ y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'}(Q(\phi_{j+1}, a'; \theta^-)) & \text{otherwise} \end{cases} \tag{2.7}$$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to network parameters θ
    Every C steps reset $\hat{Q} = Q$

DQNs offer several advantages over traditional Q-learning, making them well-suited for complex and high-dimensional environments. Their ability to approximate the Q-function allows for more efficient and scalable learning in large state-action spaces. DQNs have demonstrated remarkable success in various domains, including playing Atari games[22], robotic control, autonomous decision-making [47] and other complex tasks.

In conclusion, Q-learning and Deep Q-Networks (DQNs) constitute the fundamental pillars of the Deep Reinforcement Learning paradigm. While Q-learning provides a solid foundation for estimating Q-values and learning optimal policies, DQNs extend those capabilities by leveraging neural networks to handle high-dimensional and complex environments. The integration of DQNs into the DRL framework has revolutionized the field, enabling agents to learn sophisticated decision-making strategies in a diverse range of applications. Understanding the principles and techniques behind DQNs sets the stage for our exploration of how DRL can be applied to the context of overlapping coalition formation, as detailed in the subsequent section of the thesis.

2.1.3 Policy Gradient Methods

Policy gradient methods are an influential category within the field of Deep Reinforcement Learning (DRL). They do not suffer from many of the problems that have been marring traditional reinforcement learning approaches, such as the lack of guarantees of a value function, the intractability problem resulting from uncertain state information and the complexity arising from continuous states and actions. Unlike value-based approaches that focus on the estimation of the value of a state-action pair, policy gradient methods aim to optimize the agent's policy itself. Through this approach an agent is able to learn intricate strategies for decision-making, which are often suited to complex environments.

The objective of a Reinforcement Learning agent is to maximize the "expected" reward when following a policy π. To that end we define a set of parameters θ to parameterize this policy -$\pi\theta$. Our goal is to find the θ that maximize the expected cumulative reward $\mathbf{J}(\pi_\theta)$. This basically is the agent's objective: to learn the policy which will yield the highest expected sum of rewards over time[36]. The equation for $\mathbf{J}(\pi_\theta)$ is written as:

$$\mathbf{J}(\pi_\theta) = \mathbb{E}_{\tau\sim\pi\theta}\Big[\sum_{t=0}^{\infty}\gamma^t r_t\Big] \tag{2.8}$$

Where, π_θ is the policy parameterized by θ. The trajectory (sequence of states and actions) generated by following policy π_θ is symbolized with τ. The time step withing the trajectory is t and r^t is the reward received at time step t. Finally, γ is the discount factor, which determines the importance of future rewards relative to immediate rewards.

At the core of policy gradient methods lies the policy update rule. The REINFORCE algorithm [6], which is one of the foundational policy gradient methods, exemplifies this approach. It updates the policy by ascending the gradient of the expected cumulative reward with respect to θ. This way of gradient ascent step lead the policy towards actions that lead to a better performance and higher rewards.

The policy update rule with the use of the REINFORCE algorithm is the following:

$$\theta_{t+1} = \theta_t + \alpha \nabla \theta J(\pi_{\theta_t}) \tag{2.9}$$

Where the term θ_t is the policy parameters at time step t. The learning rate is α and $\nabla \theta J(\pi_{\theta_t})$ is the gradient of the expected cumulative reward with respect to θ.

Policy gradient methods provide various advantages. Through their direct optimization of the policy, they are able to handle discrete and continuous action spaces. That makes them quite effective at dealing with complex tasks and high-dimensional spaces. However, they are limited by cases where there is high variance in the gradient estimates, resulting to the need for techniques such as variance reduction or baseline subtraction in order to battle this issue[36].

They are also, able to address the difficult of balancing exploration with exploitation. They allow for exploration of various actions while gradually moving towards convergence on a deterministic policy, through the policy's stochasticity. They continue to get stronger with variants such as Trust Region Policy Optimization (TRPO) [26] and Proximal Policy Optimization (PPO). TRPO in order to prevent having significant policy deviations, constrict the policy updates. On the other hand, PPO by using clipped objective functions, is able to strengthen TRPO's stability, resulting in a more efficient and sample-effective learning [44].

Policy gradient methods are quite effective in multi-agent systems, since they enable agents to learn either collaborative or competitive strategies. In a setting where the actions of an agent affect the other agents' reward and the environment, policy gradients help with the learning of complex interaction dynamics and contribute to an efficient cooperation and coordination.

In conclusion, policy gradient methods are a direct approach of optimizing policies to guide agent behaviour. They are suitable for complex scenarios and are able to handle diverse action spaces. Those aspects make them a very powerful tool in the DRL toolkit. Overall, they offer us a different set of strengths and weaknesses than action-value methods[36].

2.2 Theoretical Aspects of Overlapping and Sequential Coalition Formation

In this section, we shift our focus to the theoretical aspects of overlapping coalition formation-an innovative approach that enables agents to participate in multiple coalitions simultaneously.

We start by presenting a definitive definition of overlapping coalitions and going into more detail on their distinctive characteristics in comparison to tradition non-overlapping coalitions. The concept of overlapping coalitions open the door to novel possibilities for flexible resource allocation, efficient task distribution, and enhanced knowledge sharing among agents [13]. By delving deeper, we face the challenges and benefits of forming overlapping coalitions in dynamic environments. Although the agents in overlapping coalition have increased flexibility and adaptability, at the same time they come against inherent complexities of coordination, decision-making, and potential conflicts.

By critically evaluating the theoretical challenges and benefits, we gain valuable insights into the key areas where DRL techniques can play a pivotal role. This paves the way for us to establish a strong relationship between DRL and overlapping coalition formation-two fields that, when combined, promise to offer a new way to explore the landscape of multi-agent systems and coalition formation paradigms.

2.2.1 Definition and Characteristics of Overlapping Coalitions

Coalition formation, is a paradigm used in the dynamic landscape of multi-agent systems, facilitating agents' collaboration, decision-making and resource utilization. Coalitions, emerge as dynamic assemblies of agents who collaborate in order to engage in tasks, contribute resources to achieve goals and share information. The outcomes of such goals, include task completion, resource allocation or the resolution of complex problems that would not be solvable from an individual [8].

At the heart of coalition formation lies the imperative of efficient resource sharing and task allocation. The formation of coalitions brings agents to bring their

resources, expertise and capabilities to the table and engage in coordinated decision-making. By aligning their individual interests with the collective objectives of the coalition, agents can determine the distribution of tasks, responsibilities and rewards. The practice of coalition formation is crucial in shaping the dynamics in a multi-agent system, since the overall system's performance and stability are affected by the agents' strategic choices regarding task assignments, coalition partners and resource utilization.

To better lay the theory of the coalition formation, we first need to talk about some basic game theory elements regarding coalition structures. Let's say we have a set of agents $N = \{1, \ldots, n\}$. A subset of those agents $S \subseteq N$ is a *coalition*. In order to be able to represent the value of a coalition with a single number, we need the *characteristic function*. A *characteristic function* $v : 2^N \mapsto \mathbb{R}$. A transferable utility game can be completely defined by the set of players N and the characteristic function v, so we can represent is as $G = (N, v)$.

However, we have not yet described a means of distributing the payoffs of the coalition given by the characteristic function. Say we have a vector of payoffs $x = (x_1, \ldots, x_n)$ called an *allocation* and we assign a payoff to each $j \in N$. Such an allocation x would be *efficient* with respect to a coalition structure CS, if $\sum_{j \in S} x_j = v(S)$ for all $S \in CS$ and it is called an *imputation* if it is efficient and satisfies *individual rationality*, $x_j \geq v(j)$. The set of all imputations of CS is denoted as $I(CS)$ [16].

The theoretical foundation of coalition formation is able to extend to overlapping and sequential scenarios. Overlapping coalition formation leverages the notion of agents participating in multiple coalition simultaneously, thus expanding the scope of collaboration. This approach introduces a level of dynamism and adaptability, since it allow agents to engage in diverse collaborations while leveraging their distinct capabilities across a variety of tasks [54].

In overlapping coalition formation, unlike the conventional model, agents have the ability to participate in multiple coalitions at the same time, enabling a more fluid and intricate engagement. This is a defining feature since it allows the agents to contribute to an array of tasks, share resources and interact with a broader network of peers. We can now define an overlapping coalition formation game as follows:

Definition: *An* OCF-game G *with player set* $N = \{1, \ldots, n\}$ *is given by a function* $v : [0,1]^n \hookrightarrow \mathbb{R}$ *where* $v(0^n) = 0$.

The motivation behind overlapping coalitions is multilayered. At first, agents gain more flexibility, taking advantage of their skill sets through their engagement in coalition that align with their expertise. That leads to resource allocation becoming optimized, since they allocate their efforts across multiple tasks, reducing idle resources and maximizing productivity. Additionally, overlapping coalitions allow knowledge sharing between the agents, which leads to accelerated learning.

2.2.2 Challenges and Benefits of Overlapping Coalition Formation

Overlapping coalition formation battles directly the problems of resource scarcity and underutilization that traditional coalition formation often faces[13]. Agents can distribute their resources to a wider range of tasks, making sure, that they are allocated where they are most needed. This more efficient way of resource utilization, especially shines in dynamic environments, where the availability of resources fluctuates over time. Overlapping coalition also facilitates real-time adaptation, so that the agent can respond quickly to evolving demands [54].

However, overlapping coalition formation doesn't come without its challenges. Having the agents involved in multiple coalitions can rise coordination hurdles as well as increase the overall complexity. Agents have to balance their commitments, in order for their contributions to align with the expectations of each coalition. In that way, sophisticated coordination mechanisms are needed so as to prevent potential conflicts from overlapping obligations and to ensure a stable collaboration across coalitions [23].

Other key concepts in the realm of overlapping coalition formation are the concepts of core, stability, other concepts such as the Shapley values, the kernel, and conflict resolution. The core of a coalition represents a set of payoffs where there

can be no subgroup of agents that could achieve higher payoffs by forming their own coalition. It makes sure that there are no incentives for agents to deviate from the established coalition [13]. Achieving core stability is a crucial consideration in ensuring the longevity and effective of overlapping coalition. It implies that once a coalition has been formed, no subset of agents has incentive to break away and form a new coalition, as that would result in them receiving lower payoffs [15].

The kernel of a coalition game represents a set of payoffs where no group of agents has an incentive to leave and form a new coalition. It is a subset of the core and provides a more stringent condition for stability. The kernel is particularly important in scenarios where the core might be empty, indicating potential instability [9].

The Shapley value, provides a fair way for the total payoff of a coalition to be distributed among its members. All possible permutations of agents are considered and the average marginal contribution of each agent across all possible orders is calculated. Shapley value, help us understand the distribution of gains within coalitions, and play a crucial role in determining the stability of a coalition [15].

Overlapping coalition formation stresses the importance of conflict resolution and decision-making. As agents participate in diverse coalitions, their interests conflict, thus highlighting the need for negotiation mechanisms and compromise. The potential overlap of tasks, reward distribution and resource allocation are all vital components that have to be taken into account in conflict resolution strategies, to ensure the stability and effectiveness of each coalition.[mention for conflict resolution]

Agents engaged in overlapping coalitions often have to deal with the difficulty of managing their timelines and commitments. Across multiple coalitions various elements, such as tasks, objectives and participation requirements may coincide, thus making the careful scheduling and time management a necessity. Agents have to strike a balance between the quality of their contributions without failing to fulfill their obligations in each coalition they participate [54].

Overlapping coalition formation marks the evolution of traditional coalition models, giving way to new more efficient techniques for resource efficiency, flexibility

and collaboration in multi-agent environments. The ability of agents to contribute in multiple coalitions increases their adaptability, redefining how cooperation and collective problem-solving is pursued.

2.2.3 Sequential Coalition Formation and its relation to Overlapping Coalitions

Sequential coalition formation [12] introduces a new dimension to the area of multi-agent collaboration, as it combines coalition formation with the agents' historical interactions and decision-making processes. The main difference from the traditional overlapping coalition is that it acknowledges the fact that agents do not act in isolation, but partake in a dynamic sequence of coalition formation events, where the outcome of one coalition influences the options and preferences for subsequent ones, and, importantly calls for the learning of unknown aspects of the problem, such as the coalitional values, or, importantly the "transfer learning", the unknown types (i.e private capabilities), of the participating agents. Sequential coalition formation is a specialized form of overlapping coalitions, that offers unique insights to the interaction between concurrent collaboration and temporal dynamics.

As it is natural, the concept of sequential coalition formation is connected with that of overlapping coalitions. Agents sequentially form coalitions which inherently cross into overlapping coalitions, since agents can be members of multiple coalitions simultaneously at distinct points in time. This connection further emphasizes the fact that the temporal nature of sequential coalition formation inherently gives rise to overlapping coalitions. As agents transition from one coalition to another, they introduce the potential of knowledge transfer across coalitions, resource sharing and shared participation, capturing the essence of overlapping collaboration.

The nature of sequential formation of coalitions presents us with a new layer of dynamism to overlapping coalitions. That is, since agents participate in overlapping coalition but also navigate the transitions between them. This dynamic process brought an evolution to the overlapping coalitions paradigm as agents' sequential decisions affect the composition, objectives and structure of subsequent coalitions. That sequential aspect, acts as a catalyst to the agents adapting their strategies

for participating in a formation based on their previous experiences from earlier coalitions.

2.2.4 Challenges and Benefits of Sequential Coalition Formation

There are many added challenges when sequential coalition formation comes into play. As we have previously discussed agents now have to be more considerate of the temporal alignment of tasks, the overall impact on the cumulative reward from participation decisions, the fact that the cooperation values between the types are unknown, as well as the fact that earlier coalitions affect later ones, which raises the need for a feedback loop and finally the amplified complexity of resource allocation [11]. The task of managing the sequence of the coalition events requires robust mechanisms to handle the aforementioned problems. These challenges highlight the complex nature of handling dynamic, overlapping collaborations in a sequential manner.

Although the above challenges may make sequential coalition seem more bleak and overwhelming with challenges, that is not the complete picture, since it has a variety of strengths that help us deal with the above difficulties. Sequential coalition formation gives agent the ability to adapt and learn from the coalitions they participate in. They gain insights into optimal task assignments, cooperative behaviour through successive interactions and effective strategies. Also, as agents join, leave and form new coalitions, the objectives of coalitions adapt to changing circumstances. This adaptability greatly strengthens agents' decision-making prowess while allowing them to refine their participation choices over time, as well as, augment coalition stability and empower effective responses to shifting environmental conditions.

Additionally, it provides the agents with encounters of diverse peers, resources, and tasks, encouraging knowledge sharing. This dynamic collaboration is beneficial to enhancing creativity and the exploration of problem-solving approaches. On top of that, agents through this sequential manner of forming coalitions, learn to allocate their resources strategically over different tasks and interactions [19]. This refined

resource utilization secures that agents contribute where their capabilities have the highest impact, thus improving the overall efficiency.

2.2.5 Synergy between Overlapping and Sequential Coalition Formation

The synergy between Overlapping and Sequential coalition formation is evident in their mutual reinforcement. Sequential coalition formation infuses the concept of time into overlapping coalitions, adding to them a narrative of progression and adaptation. On the other hand, overlapping coalitions improve the sequential aspect through giving the agents the ability to leverage diverse experiences and resources across coalition, enriching their decision-making in subsequent interactions.

In embracing sequential coalition formation and its relationship with overlapping coalitions, we endeavor to find out more about the intricacies of multi-agent cooperation in dynamic environments. To support this journey, we use various tools to design innovative algorithms, strategies and protocols in order to optimize our agents' participation decisions across time and collaboration limits. By delving into synergy between sequential and overlapping coalition formation, we aim to unlock the potential to enhance efficiency, adaptability, and collective performance of multi-agent systems that face complex challenges and evolving objectives.

2.2.6 The role of DRL in Overlapping and Sequential Coalition Formation

Deep Reinforcement Learning (DRL) potentially offers a powerful tool in dealing with the intricacies of both overlapping and sequential coalition formation. As such, DRL takes advantage of the principles of reinforcement learning and alongside deep neural networks, plays a crucial role in addressing challenges, optimizing agents' strategies and harnessing opportunities[62].

This is because, DRL empowers agents to learn intricate decision-making strategies required for navigating overlapping and sequential coalitions. Agents gain the ability to evaluate the benefits and drawbacks of sequential coalition engagements,

adapt their participation decisions based on previous interactions, and dynamically adjust their strategies to evolving circumstances [61]. Overlapping and sequential coalition formation demand a delicate balance between exploration and exploitation [11]. DRL algorithms incorporate exploration-exploitation trade-offs, like $\epsilon - greedy$ policy, enabling agents to explore alternative coalition configurations while capitalizing on well-performing ones [22].

Moreover, Overlapping and sequential coalition formation occur in dynamic and evolving environments, with multiple agents, coalitions and temporal dynamics, where the state and action spaces can be vast and complex [61]. DRL's utilization of deep neural networks, allows agents to process and represent high-dimensional data efficiently, while DRL's adaptability allows them to respond in real-time to changing circumstances, adjusting their strategies according to changes in task demands, resource availability and coalition dynamics.

DRL's reinforcement learning framework enables agents to determine resource allocations that lead to the highest cumulative reward over times, as well as, optimize agents' participation decisions contributing to coalition stability [62]. Additionally, is has the capacity to integrate past coalition participation outcomes in the learning process, equipping the agents with the ability to make informed decisions that account for the long-term implications of their participation choices.

In essence, DRL can contribute to the advancement of overlapping and sequential coalition formation research. It empowers agents to navigate through the various challenges and dilemmas we have mentioned so far, helping to move the field towards more intelligent, adaptive and efficient multi-agent systems. As DRL algorithms continue to evolve and adapt, so does their potential to change the landscape of both overlapping and sequential coalition formation, unlocking new frontiers in decision-making and collaboration in such dynamic and complex settings.

2.3 Fundamentals of Graph Neural Networks (GNNs) and extensions

In this section, we aim to present the basic concepts of Graph Neural Networks(GNNs) [50], [56], which is a class of artificial neural networks for processing data represented in the form of graphs. Their versatility offers distinct advantages, particularly when dealing with substantial datasets, which can be vital to our DRL-overlapping coalition formation approach. That is, we envisage these networks to play a pivotal role in mitigating the uncertainty agents face when forming overlapping coalitions regarding the co-operation values between the various agent types.

As we will explain on chapter 4 in greater detail, our approach uses GNNs to update those values, by transforming each coalition to a graph, and by using an extension of GNNs, called Graph Attention Networks [50], to update the co-operation values between the types in a different and more scalable way, especially when we deal with a huge agent set, forming a big number of overlapping coalitions per episode. This dynamic adjustment ensures precise and adaptable cooperation values based on coalition interactions [59].

2.3.1 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of machine learning models designed to operate on graph-structured data. Unlike traditional neural networks that process grid-structured data like images or sequences, GNNs can effectively handle complex relationships and dependencies inherent in graph data [50].

In a graph, data is represented as nodes (entities) connected by edges (relationships or interactions). This structure is common in various domains, such as social networks, recommendation systems, biological networks, and more. GNNs leverage this structure to learn and make predictions about the nodes or the entire graph.

The key idea behind GNNs is message passing. At each step, nodes exchange information (messages) with their neighbors. This process allows nodes to aggregate information from their local neighborhood, enabling them to make informed decisions about their own properties or the properties of neighboring nodes [56].

Graph Neural Networks (GNNs) enables one to harness the inherent structure of graphs when processing information. As mentioned above, our approach leverages GNNs to update co-operation values between agents types. This is achieved through transforming each coalition into a graph representation, thereby enabling GNNs to dynamically adjust co-operation weights based on coalition interactions. This integration of GNNs ensures that agents' co-operation values change according to their experiences, enhancing the precision and adaptability of overlapping coalition formation.

One of the most prominent features of GNNs, is their ability to incorporate contextual information from neighboring nodes [50]. In the context of sequential overlapping coalition formation, this implies that an agent's co-operation values are influenced not only by its direct interaction but also by the interactions of its coalition partners. Moreover, GNNs introduce a layer of adaptability and learning into co-operation value updates. As agents engage in diverse coalition interactions over time, GNNs enable them to gain insights from there experiences and adjust their co-operation values accordingly.

However, as it is to be expected it faces its own set of challenges and drawbacks. Understanding them is crucial in our implementation of them in optimizing the performance of our update model. GNNs computational complexity can escalate with the size of the graph, leading to potential scalability challenges when dealing with large agent sets or extensive coalition formations.The inherent risk of overfitting also arises when GNNs adapt too closely to the training data, potentially leading to poor generalization to unseen coalition formations.

GNNs can become intricate and challenging to interpret, especially when multiple layers and attention mechanisms are incorporated[37]. Also, their performance is contingent on hyperparameter settings, including the number of layers, attention mechanisms, and learning rates. They face as well, challenges in scenarios where limited historical data is available for new or rarely encountered coalition formations. Temporal dynamics is another element that they may not fully capture in sequential coalition formation due to the fact that GNNs inherently assume a static graphic structure [50].

In conclusion, while GNNs offer a promising avenue for updating co-operation

values within sequential overlapping coalitions, acknowledging and addressing there challenges is crucial. By carefully navigating scalability concerns, information propagation variations, overfitting risks, model complexity, and temporal dynamics [59], we can enhance the robustness, effectiveness, and adaptability of GNN-based approaches in the domain of DRL-based sequential overlapping coalition formation.

2.3.2 Graph Attention Networks

A noteworthy extension of GNNs, Graph Attention Networks (GATs) [37], [50], emerge as a neural network architecture for processing graph-structured data. Whether is is for modeling social networks, molecular structures, or language syntax, GATs leverage attention mechanisms to weight the importance of the different nodes and edges in a graph.

The attention mechanism used in Graph Attention Networks (GAT) is a technique that allows nodes in a graph to dynamically weigh the importance of information received from their neighbors during message passing. This weighting is learned from the data itself.

In GAT, each edge connecting two nodes is associated with an attention score. These scores are computed based on a learned weight matrix, which is trained alongside the rest of the neural network. The attention scores reflect how much attention each neighbor should receive when sending messages to a particular node [37], [48].

This capability allows them to concentrate on the most relevant information, thus facilitating more informed and accurate predictions. The versatility of GATs is underscored by their application in diverse tasks such as node classification, link prediction, and graph generation[50].

GATs stand as an evolution of Graph Convolutional Networks (GCNs) [50] [56] introducing an innovative approach to graph-based learning. Their exemplary performance across benchmark datasets, like CORA, has established GATs as state-of-the-art solutions in various domains. The inherent ability of GATs to capture intricate dependencies within graph structures aligns seamlessly with out pursuit of optimizing co-operation values in sequential overlapping coalition formation [48].

GATs operate on fundamental matrices essential to their functioning. These include the *adjacency matrix*$_{\alpha_{ij}}$ of dimensions $[nodes \times nodes]$, the $feature\ matrix$ of dimensions $[nodes \times features]$, the *weight matrix* of dimensions $[features \times 2 \cdot features]$ and finally the matrix for the *embeddings per node* of dimensions $[nodes \times 2 \cdot features]$. These foundational elements pave the way for GATs to comprehend intricate relationships and dependencies within graph-structured data.

In the context of our approach, GAT are harnessed to process coalitions represented as graph structures. When presented with a coalition, GATs generate attention coefficients that are vital to the update of the co-operation values between diverse types of agents. The feature assignment to each node of the graph is designed to encompass the type of agents represented, the count of agents of the corresponding type, and the cumulative reward acquired within that coalition.

Below is an example of how a graph where the nodes represent the types of agents would look like:

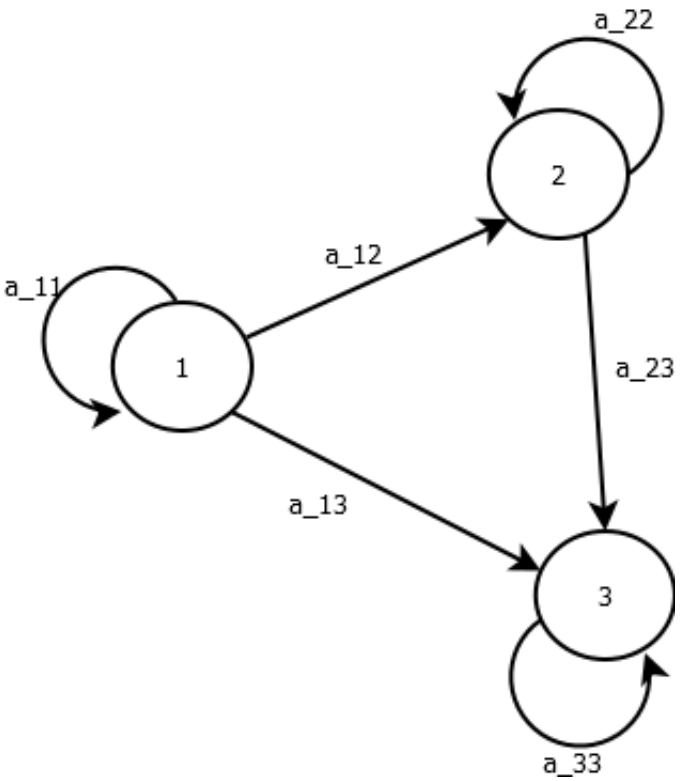

Figure 2.2: Attention Graph example

The relationships between these matrices are elegantly encapsulated in the following formula, as described in [37]:

$$h_i = \sigma\left(\sum_{j \epsilon N(i)} \alpha_{ij} W h_j\right) \tag{2.10}$$

The h_i component represents the output (hidden state) of node i after the message passing step. The σ denotes the activation function applied element-wise to the weighted sum. Usually it is the sigmoid function, ReLu or Leaky ReLu, in our case we will use the latter [37]. The sum $\sum_{j \epsilon N(i)}$ stands for the summation over all nodes j that are neighbors of node i in the graph. $N(i)$ represents the set of neighbors of node i. The attention coefficient α_{ij} is associated with the edge from node i to node j. These coefficients are learned during training and determine the importance of node j to node i. The learned weight matrix mentioned before is represented by W and is applied to the hidden states of the neighbor nodes. It allows the model to learn the relationships between nodes. Lastly, h_j is the hidden state of node j (neighbor node) before message passing step.

To summarize, this formula describes how the hidden state of node i is computed by aggregating information from its neighbors j according to learned attention coefficients a_{ij}. The information from each neighbor j is weighted by W and combined using the activation function σ to produce output h_i.

This operation serves as guidance for our agents, as they navigate the graph's intricate architecture to discover the co-operation values among agent types, which is the very essence of GAT's attention coefficients. Armed with this understanding, out agent delve into the realm of Deep Reinforcement Learning (DRL), applying the acquired knowledge to task completion and sequential overlapping coalition formation.

Following, the learned attention coefficients are harmoniously propagated back to the GAT. This feedback loop is vital for the iterative enhancement of co-operation values updates, a dynamic process that underscores the synergy between GAT and

DRL for our sequential overlapping coalition formation approach.

By incorporating Graph Neural Networks and their GAT extension in our DRL framework, we embark on a journey to imbue our sequential overlapping coalition formation process with adaptive learning, structured information processing, and improved scalability. This integration supplies our approach with the capacity to dynamically enhance agents' coalition participation decisions through evolving co-operational values, ultimately enhancing the efficacy and adaptability of multi-agent collaboration.

Chapter 3

Related Work

This chapter delves into the diverse landscape of research and studies that collectively inform and contextualize the present work on Deep Reinforcement Learning(DRL)-based overlapping and sequential coalition formation. The purpose of this segment is to serve as a compass, navigating through existing literature and methodologies in the above fields, that helped us shape our approach and environment.

In this subsequent sections, we explore the realms of DRL in multi-agent systems, coalition formation and cooperative games, the power of Graph Neural Networks(GNNs) and Graph Attention Networks(GATs), as well as, the intricacies of sequential decision-making and overlapping coalitions. Through this elaborate exploration, we try to identify gaps in current knowledge and by the contributions of our research, try to fill those gaps.

3.1 DRL in Multi-Agent Systems

In the extensive field of Deep Reinforcement Learning (DRL), a paper titled "A Survey and Critique of Multi-Agent Deep Reinforcement Learning" by Hernandez-Leal et al, stands as a guidepost for understanding the intricacies of multi-agent systems in the context of DRL [40]. The authors begin by presenting the unique challenges that underlie multi-agent scenarios, from issues of coordination and competition to communication and exploration, which mirror the complexities inherent in our research context.

Of special interest is the paper's critique of existing DRL methodologies. The

authors assess the strengths and weaknesses of various approaches, highlighting key considerations for choosing appropriate techniques within multi-agent settings.

Furthermore, the paper delves into the challenges of generalization, transfer learning, and scalability in the realm of DRL, addressing concerns relevant to the application of our research in real-world scenarios. By navigating these issues, the paper provides insights into adapting DRL techniques to complex and evolving multi-agent environments, mirroring the challenges we seek to address in this thesis.

In essence, "A Survey and Critique of Multi-Agent Deep Reinforcement Learning" bridges the gap between DRL and multi-agent systems, complementing our thesis's exploration of coalition formation. It establishes a solid connection between our specialized research and the broader landscape of DRL, fortifying the significance of our work within this dynamic and rapidly evolving field.

In the realm of deep reinforcement learning (DRL) and multi-agent systems, the paper "Multi-agent Deep Reinforcement Learning for Task Allocation in Dynamic Environment" [31] makes an important contribution to the dynamic landscape of task allocation. Authored by Noureddine, Gharbi, Ahmed and Samir, this paper navigates the intersection of multi-agent coordination, dynamic environments, and deep reinforcement learning—a subject of paramount importance in our thesis's exploration of sequential overlapping coalition formation.

One of the key strengths of this paper is its integration of deep reinforcement learning techniques into the task allocation problem. It explores how agents can adapt and make decisions in real-time, a crucial aspect of sequential coalition formation.

Furthermore, the paper presents a comprehensive evaluation of the proposed multi-agent DRL framework, shedding light on its efficacy and performance in dynamic scenarios. The evaluation metrics and methodologies employed can provide valuable insights into assessing the success and stability of coalition formation strategies in our research context.

In summary, "Multi-agent Deep Reinforcement Learning for Task Allocation in Dynamic Environment" provides a bridge between DRL, multi-agent systems, and

dynamic task allocation—an intersection central to our thesis. Not only it strengthens the foundation of our research but also showcases the practical relevance of our work in addressing the complexities of sequential overlapping coalition formation.

3.2 Coalition Formation and Cooperative Games

The intricate domain of coalition formation within uncertain multi-agent systems has spurred significant research efforts to devise effective decision-making strategies. One noteworthy contribution in this realm is the paper "Bayesian Reinforcement Learning for Coalition Formation under Uncertainty" authored by Boutilier and Chalkiadakis [10]. They recognize that in real-world scenarios, coalition formation occurs amidst incomplete and noisy information, making traditional approaches inadequate. To tackle this issue, they propose a model-based Bayesian reinforcement learning framework that integrates uncertainty modeling, learning, and coalition formation dynamics. While in our approach we try a model-free with some model-based elements framework.

A pivotal feature of their framework is the utilization of Bayesian inference to estimate the uncertain parameters of the MDP model. By leveraging prior beliefs and observed outcomes, agents can update their knowledge about the environment and the coalition formation process. This adaptive learning mechanism equips agents to make more informed decisions over time, thus enhancing the quality of coalition formations.

Importantly, the paper thoroughly explores the concept of the "core" in coalition formation games, a key stability notion. They establish how Bayesian reinforcement learning, by providing agents with accurate uncertainty estimates, contributes to ensuring that coalition outcomes lie within the core. This not only improves the stability of coalitions but also ensures that formed coalitions are more resistant to deviations.

Furthermore, the work emphasizes the importance of incorporating domain-specific knowledge into the Bayesian reinforcement learning framework. By tailoring the prior distributions and likelihood functions to the specific characteristics of the

coalition formation problem, agents can exploit their domain expertise to enhance the learning process. This feature resonates strongly with the real-world applicability of the approach.

The paper also represents a comprehensive framework for Bayesian reinforcement learning in coalition formation, as well as, allowing agents to form coalition while adapting to the evolving environment. Their approach enables agents to learn policies that take into account the uncertainties in the coalition formation outcomes. Leading to the agents forming more refined strategies for coalition participation.

In summary, "Bayesian Reinforcement Learning for Coalition Formation under Uncertainty" provides a valuable contribution to the understanding of how Bayesian methods can synergistically merge with reinforcement learning techniques to tackle the intricate challenges posed by coalition formation in uncertain multi-agent environments. The incorporation of domain-specific knowledge, the exploration of the core concept, and the demonstrated efficacy of their approach make this work a pivotal reference for designing robust and adaptive coalition formation strategies.

The paper "A cooperative game-theoretic approach to the social ridesharing problem" by Chalkiadakis et al. [29] offers insights to the application of cooperative game theory to ridesharing. It introduces a game-theoretic framework that addresses social aspects of ridesharing by taking into consideration the collaborative dynamics among participants.

The paper's contributions lie in formalizing the ridesharing problem as a cooperative game, accounting for the interests of both drivers and passengers. It introduces mechanisms to allocate costs fairly and incentivize cooperative behavior, underscoring the importance of collaboration in ridesharing scenarios.

While distinct in application, the paper's insights align with our DRL-based overlapping coalition formation. Chalkiadakis et al.'s work showcases the significance of cooperative behavior in multi-agent systems, reflecting the collaborative essence of our research.

The study's application of cooperative game theory to ridesharing resonates with our exploration of coalitions formed for mutual benefit. By adapting insights from this framework, we can enhance our approach's co-operation value updates and

ensure equitable outcomes in overlapping coalition contexts. It addresses the challenge of fairly allocating costs, a parallel to our objective of updating co-operation values. Their mechanisms to incentivize cooperation offer inspiration for fostering collaborative dynamics among agent types.

In summary, [29] bridges cooperative game theory and ridesharing challenges. While centered on a different application, its contributions underline the significance of collaborative frameworks, resonating with our DRL-based overlapping coalition formation approach. The insights from this paper inform our endeavor to navigate multi-agent collaboration within complex coalitions.

3.3 GNNs and GATs

Graph Attention Networks (GATs) [37] are a groundbreaking innovation in graph neural networks. Developed by Velickovic et al., GATs introduce a novel self-attention mechanism that revolutionizes information propagation in graph-structured data.

GATs address the limitations of traditional graph convolutional networks(GCNs) by enabling nodes to adaptively focus on influential neighbors. This attention mechanism enhances context-aware feature propagation and precise predictions within graphs. GATs align with our DRL-based overlapping coalition formation approach. By transforming coalitions into graph structures and employing GATs, we can dynamically adjust co-operation values based on coalition interactions and contextual agent participation. In sequential coalition formation, GATs can potentially capture evolving relationships and temporal dynamics, as we envisage in our work for the first time in the literature. Their attention mechanisms accommodate changing agent interactions, crucial for dynamic sequential coalition formations.

To summarize, [37] significantly advances graph neural networks and aligns perfectly with our DRL-based overlapping and sequential coalition formation goals. GATs enhance co-operation value updates, accommodate evolving coalitions, and fortify multi-agent collaboration precision. This pivotal paper provides a strong foundation for our fusion of DRL, GATs, and coalition formation.

In the field of machine learning, graph neural networks (GNNs) have gained significant attention for their applicability in modeling complex data structures. The paper "Graph Neural Networks: A Review of Methods and Applications" [50] serves as a comprehensive and influential contribution to the domain of GNNs. Given our exploration of deep reinforcement learning (DRL) in multi-agent systems, which often involve intricate interactions represented as graphs, its contribution to our case is noteworthy.

This paper's significance to our research lies in its meticulous review of GNN methods and their diverse applications. Multi-agent systems inherently possess graph-like structures, where agents and their relationships can be represented as nodes and edges in a graph. It delves into various GNN architectures, including Graph Convolutional Networks (GCNs), GraphSAGE, and Gated Graph Neural Networks (GGNNs), among others. This knowledge can be invaluable in our exploration of deep reinforcement learning algorithms, as GNNs often serve as a foundational component in modeling multi-agent interactions.

Furthermore, the review provides insights into the applications of GNNs across a wide spectrum of domains, ranging from social network analysis to recommendation systems and molecular chemistry. These diverse applications parallel the multi-faceted nature of multi-agent systems, where agents collaborate, compete, and form coalitions for various purposes.

To summarize, "Graph Neural Networks: A Review of Methods and Applications" provides useful insights to the world of GNNs and their applications. It helps us better understand the GNNs for the powerful tool for modeling and analyzing complex relationships and dependencies among agents in our research context that can be represented in graph-like structures. Additionally, it guides us in identifying areas where DRL algorithms and GNNs can synergize to address complex multi-agent problems.

3.4 Sequential Decision-Making and Overlapping Coalitions

In the domain of sequential overlapping coalition formation, an influential paper titled "Sequential Decision Making in Repeated Coalition Formation under Uncertainty" by Chalkiadakis et al. has made a significant contribution [12].

The paper presents a novel framework that combines sequential decision-making with coalition formation under uncertainty. The framework involves agents making decisions over time to create coalitions, considering the evolving environment and the uncertain outcomes of their actions. This sequential approach is highly suitable for situations where agents need to adjust their coalition strategies based on changing circumstances, enabling optimal decision-making over time.

A notable contribution of the paper lies in the formulation of a dynamic Markov Decision Process (MDP) to model the sequential coalition formation problem. Chalkiadakis incorporates uncertain transition probabilities, which capture the uncertainty in coalition formation outcomes, into the MDP framework. This enables agents to explicitly consider uncertainty when making decisions, leading to more robust and adaptable coalition formation strategies.

The insights derived from "Sequential Decision Making in Repeated Coalition Formation under Uncertainty" paper offer a valuable groundwork for comprehending the challenges and strategies associated with sequential coalition formation under uncertainty. The innovative framework and algorithms proposed in the paper open avenues for further research in designing efficient and effective approaches for multi-agent systems that confront evolving coalition dynamics and uncertain outcomes.

"Overlapping Coalition Formation" by Chalkiadakis et al. [13] presents an influential exploration into the realm of multi-agent cooperation through overlapping coalitions, a topic fundamental to our study.

Chalkiadakis et al. delve into the concept of overlapping coalition where agents can participate in multiple coalitions simultaneously. The paper's contributions lie in formalizing overlapping coalition structures. It introduces notions of stability,

efficiency, and fairness in the context of multi-coalition participation. These insights resonate with our focus on DRL-based overlapping coalition formation. Chalkiadakis et al. work provides a foundational understanding of the challenges and benefits associated with overlapping coalitions.

The study's emphasis on stability and fair reward distribution mechanisms, aligns with our aim to optimize cooperation in overlapping coalitions. Their insights guide our approach's consideration of fairness when updating co-operation values, fostering balanced collaboration among diverse agent types.

"Overlapping Coalition Formation" by Chalkiadakis et al. stands as a pivotal work that introduces and explores the dynamics of overlapping coalitions. Its contributions align seamlessly with our goal of DRL-based overlapping coalition formation, informing our understanding of coalition stability, equitable reward distribution, and multi-agent cooperation in complex scenarios.

The paper entitled "Overlapping Coalition Formation via Probabilistic Topic Modeling" by Chalkiadakis and Mamakos [35] has deeply influenced and initiated this thesis' research in the domain of overlapping coalition formation. Its importance to our research lies in its innovative approach to addressing a fundamental challenge in multi-agent systems: how agents autonomously and strategically form coalitions while potentially belonging to multiple coalitions simultaneously. This idea significantly broadens the applicability of coalition formation in real-world scenarios, aligning with the complex and dynamic nature of multi-agent systems. The introduction of this concept was instrumental in shaping the direction of our thesis, as it allowed us to explore novel and realistic scenarios in our research.

The paper leverages probabilistic topic modeling, specifically Latent Dirichlet Allocation (LDA), as a framework to model agents' interests and topics of cooperation. This innovative use of topic modeling techniques demonstrates the versatility of machine learning approaches in addressing multi-agent coordination problems. This modeling technique's applicability extends beyond the paper's context and can be a valuable tool for capturing agents' behavior and preferences in overlapping coalitions.

Chalkiadakis and Mamakos address the challenge of scalability by proposing a

dynamic programming algorithm for coalition formation in large multi-agent systems. This practical approach aligns with our thesis's focus on scalability, as we explore the application of deep reinforcement learning algorithms in similar scenarios. The paper's insights into handling large agent populations helps our research in designing efficient and effective algorithms for coalition formation.

"Overlapping Coalition Formation via Probabilistic Topic Modeling" laid the conceptual and methodological groundwork for this thesis. It inspired us to investigate how deep reinforcement learning techniques, such as Q-learning and policy gradient methods, can be applied to optimize agents' participation decisions in overlapping coalitions. The pioneering nature of this paper made it a catalyst for our research journey, and its concepts continue to shape the core of our work.

Chapter 4

Our Approach

4.1 Basics

We consider the environment to be populated by a set of *Agents* A where $A = [A_1, A_2, ..., A_N]$ each of which possess a discrete, integer quantity of resources denoted by $r_i \, \epsilon \, \mathbb{N}$. Their resources are pivotal in proposing and participating in coalitions $\mathcal{C} \in \mathscr{C}$, where $\mathscr{C}$ denotes the space of all possible coalitions. This dynamic enables agents to strategically leverage their resources for collaborative endeavors.

Each *agent* is characterized by a distinct *type* represented as $t\epsilon\{t_1, t_2, \ldots, t_n\}$, where $n \in \mathbf{N}$, and in general $n << \mathbf{N}$ (with $\mathbf{N}$ being the total number of agents). This typology affects the cooperation values, referred to as Degree of Cooperation (DoC), that signifies the value of cooperation of an agent engaging in a coalition C with agents of different types, that is: $DoC(\overrightarrow{t_C}) : \times_i t_i \rightarrow \mathbb{R}, i \in \mathcal{C}$

The $Tasks$ (T) are represented as a set $\{T_1, T_2, \ldots, T_n\}, n \in \mathbb{N}$, each with specific resource requirements for successful execution. The rewards yielded from the completion of a task are dependent on the types composition of the coalition that completed the task.

Specifically, a coalition C acquires reward $R(T_C)$ for completing a task T_C, with $R(T_C) : \times_i t_i \rightarrow \mathbb{R}$, where t_i refers to the type of each agent in C (i.e., $\overrightarrow{t_C} = \langle t_i \rangle, i \in C$). Then, $R(t_C)$ is divided among the agents in C in some way. In our experiments, it is divided equally to the participating agents of coalition C that completed task

T_C. If however, the coalition C is unable to complete task T_C, it receives a penalty equal to the reward $R(t_C)$, which is also divided equally among the agents of C.

This dynamic interplay, of the task's requirements alongside the coalition dependent reward generation, underscores the necessity for resourceful coalition formations, as well as, reinforcing the strategic significance of forming coalition with compatible types for specific tasks.

Notably, each agent type contributes a predetermined integer amount of resources upon joining a coalition. This provision of resources by agents types serves as a fundamental element in the coalition formation dynamics and influences the resource availability for task execution.

So, the reward generation is a function of the DoC, which is a distinction of our work from previous model-based methods [19].

In our case, we do not know the true DoC and make updates regarding it values, while in [19] it was known because of the mentioned [19] transition dynamics $Pr(s|t_c, a_c)$s denoting the probability of outcomes given that coalition C whose members have type vector t_C take coalitional action α_C. In that case, the types were not known, while we can observe the types but are not sure for the cooperation values of them.

Now, Graph Attention Networks ($GATs$) [37] serve as a pivotal tool in encapsulating the intricacies of inter-type cooperation. Within this framework, agent types are represented as nodes, and the edges interconnecting them embody the cooperative relationships, weighted by their respective degrees of cooperation. The attention coefficients, computed by the GAT, provide the agents with crucial insights into the strengths and implications of their inter-type cooperation. This network driven intelligence can potentially empower agents to navigate the environment of sequential overlapping coalition formation with informed decision-making (and, as shown in our experiments, it manages to do so).

We propose in our approach, that the agents use ($Deep$) $Reinforcement\ Learning$ ($(D)RL$) [61] to take formation decisions; and they can be informed in their deliberation via the use of GATs (for the first time in literature), or by other means, to

update their estimates of the degrees of cooperation DoC. As such agents are able to proficiently employ RL/DRL independently and further augment their capabilities by leveraging GATs to gain additional insights into the Degrees of Cooperation (DoC).

In our approach, $GATs$ can play a pivotal role in providing agents with valuable insights into the intricate dynamics of cooperation among different agent types, encapsulated by attention coefficients. Subsequently, armed with this knowledge, our agents engage in RL/DRL protocols tailored to these cooperative nuances. Through this process, they are able to adeptly navigate the complexities of task completion and coalition formation.

Following, the conclusion of the RL/DRL *Sequential Overlapping Coalition Formation* phase (SOCF), agents share their acquired insights back to the GAT, contributing to the enrichment of its knowledge repository. This two-tiered approach ensures that agents benefit from the advantages of both RL/DRL and GAT, fostering a synergistic relationship between the two methodologies.

4.1.1 Setting up the learning environment

A vital part of our Sequential Overlapping Coalition Formation environment, is none other than the state and action representation for our algorithms. For the state (of the environment) as perceived by the decision making agent controlling its own MDP, we use its remaining resources, r, where $r \in \mathbb{N}$, and its until that point formed coalitions.

As for the action representation, it consists of a coalition formation C, comprised of a combination of the agent types of any length between 2 and $nTypes$, and a task T from the list of tasks, which has not yet been completed.

Coalition rewards upon task completion are meticulously crafted to follow a Gaussian distribution centered around a distinctive mean. This pivotal mean varies

based on the amalgamation of agent types within the coalition, reflecting the unique contributions and synergies each type offers. We encapsulate these mean rewards within a matrix, where rows denote agent types, and columns represent coalition types (comprising exclusively of combinations of the agents types of length 2).

This matrix undergoes customization for each task, mirroring the specific requirements for agent type combinations most conducive to task achievement. To facilitate this, we develop a mapping function, adept at associating each conceivable set of agent types with the appropriate mean from the distribution, thereby generating task-specific rewards. In more detail, the reward function will be explained on the following chapter, as it is a fundamental aspect of the experimental setup and update of the cooperation values between the agent types DoC.

In pursuit of realism akin to real-world scenarios, we introduce a layer of uncertainty into the environment. This is accomplished by integrating an ϵ-greedy policy for decision-making. This policy empowers the responding agent to opt for the action with the highest Q-value with a probability of $(1 - \epsilon)$ while introducing a calculated level of randomness with a probability of ϵ.

Another avenue for injecting uncertainty lies in employing a stochastic policy to designate the next proposer. One effective approach is utilizing a softmax distribution with temperature scaling [49], [24], [30]. This distribution furnishes probabilities for each agent to assume the role of proposer. Notably, agents with recent proposer roles are afforded lower probabilities, introducing an element of unpredictability and dynamism to the proposal process.

The temperature parameter controls the degree of randomness in the distribution, with higher values leading to more exploration and lower values leading to more exploitation. By decreasing the temperature over time, the distribution becomes more focused and less random, making it harder for the same agent to become the proposer repeatedly. The softmax function with the temperature parameter is as follows:

$$s_T = \frac{exp(z_i/T)}{\sum_j exp(z_i/T)} \tag{4.1}$$

The scale of temperature controls the smoothness of the output distribution. It, therefore, increases the sensitivity to low-probability candidates. As $T \to \infty$, the distribution becomes more uniform, thus increasing the uncertainty. Contrarily, when $T \to 0$, the distribution collapses to a point mass.

4.2 (Deep)RL-based Coalition Formation

Our coalition formation process integrates Deep Reinforcement Learning (DRL) in a strategic manner. Each agent can potentially leverage insights from the Graph Attention Network (GAT), where nodes represent agent types and edges encapsulate the cooperative values of an agent type with other types, or use a simple update protocol to do so, without the use of GAT.

4.2.1 DRL for Sequential Overlapping Coalition Formation

At the beginning of a round, a proposer is randomly selected (See in Algorithm 3). This agent assumes the pivotal role of suggesting coalition formations to other agents for task fulfillment. Employing a DRL algorithm, the proposer orchestrates the formation of coalitions, aiming to maximize rewards reaped from successful task completions.

The proposer's authority is bound by a pre-defined integer proposal limit, denoted as k, where $k \in \mathbf{N}$. To acquire rewards, a coalition must be formed and a task has to be successfully executed. Failing to meet the task's requirements, will lead to the members of the coalition receiving a penalty, while having a coalition proposition rejected leads to zero rewards for the proposer, for that suggestion, and he has to make another in its place.

Upon receiving a coalition proposal, the prospective coalition member initiates their own DRL algorithm. This enables them to evaluate the potential benefits of the proposed coalition in relation to their strategic objectives. If the proposed formation aligns with their perceived as best coalition configurations, or meets a predefined percentage threshold of similarity (e.g., 75 %), they accept. The similarity is calculated by comparing the proposed type vector $t_{proposal}$ with each of the perceived as best type vectors t_{best}, considering the presence and frequency of each type. Formally, the similarity between two type vectors A and B, with taking into account the frequency of each type in the vectors, can be expressed using the Jaccard similarity coefficient [1] with duplicates.

Let $\text{freq}(x, A)$ represent the frequency of element x in set A. Then, the Jaccard

similarity with duplicates ($J_d(A, B)$) can be expressed as:

$$J_d(A, B) = \frac{\sum_{x \in A \cap B} \min(\text{freq}(x, A), \text{freq}(x, B))}{\sum_{x \in A \cup B} \max(\text{freq}(x, A), \text{freq}(x, B))} \tag{4.2}$$

In this formula, $A \cap B$ represents the intersection of sets A and B, and $A \cup B$ represents the union of sets A and B. This way, if a type appears multiple times in one vector but only once in the other, it would not heavily impact the similarity calculation.

For instance, if one of the most advantageous coalitions for the respondent has a type vector t_{best} consisting of {type1, type3, type4, type5}, and the proposed formation has a type vector $t_{proposal}$ with {type5, type4, type1, type7}, the threshold criterion would be met, so the responder would accept the proposal.

An important clarification to avoid confusion is the definitions of round and episode in our setting. A round is marked by the random choice of an agent as the proposer and it is concluded when it has done k accepted proposals, after which a new proposer will be chosen, marking the beginning of a new round.

An episode, however, consists of many rounds, and it starts with resetting the environment. Meaning, to reset the agents' rewards, their proposal counter, and set the tasks' completed status to false. Subsequently, rounds, like the one mentioned above, occur until a terminal condition is met, on which we will elaborate further down, leading to the end of an episode, where the replenishing of the agents' resources and dissolving the formed coalitions takes place.

Algorithm 3 DRL coalition formation stage - Update at the end of the episode

Require: Initialize set of agents, A
Initialize the list of Tasks, T
Initialize the degrees of cooperation between the agents types DoC
for each episode **do**
Reset the environment
while no Terminal condition has been met **do**
An agent i is randomly selected as the proposer
Proposer i uses RL/DRL algorithm in order to
make k proposals using ϵ - greedy policy
The propositions are sent to the relative agents
Each responding agent in turn uses RL/DRL algorithm to respond
if proposal is accepted **then**
Coalition C is formed to complete task T_C
The reward/penalty R for completing T_C is
distributed equally among the participating agents.
else
Another proposition is made in its place from the proposer i
Update the degrees of cooperation (DoC) between the types
using simple/GNN update protocol
Resolve the formed coalitions
Replenish the resources of every agent

Another important element of our approach, that we will elaborate on Chapter 5, concerns the temporal dynamic of updating the degrees of cooperation (DoC) between the agent types. As it shown above (See in Algorithm 3) it happens at the end of an episode, but we will also test the case where the aforementioned update occurs after the end of the proposer's turn. In that way, a sense of realism is added since the decisions our agents will make would have a more informed DoC at hand.

The terminal conditions described above mark a state of our environment where the coalition formation process cannot continue. Such a condition is the case if there are not any tasks left that have not been successfully completed, that means that an additional coalition cannot be formed since there is not task for it to achieve. Also, having every agent become a proposer once marks a terminal condition in our case.

Another case is when the environment's agents run out of, or have insufficient resources to their disposal in which scenario they 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 run, to avoid over-saturation.

In the above manner, an agent partakes in coalition formation sequentially which can lead to overlapping coalitions being formed, where the repeated coalition formation as a protocol was inspired from the work of [12].

Our goal from this *training* is for the agents to learn which coalition formations are the most beneficial for them to form, from the ones they propose/are proposed in order to acquire the most reward. The RL/DRL algorithm will run with one round being an *agent* proposing *coalitions* until its k propositions are processed, with $k \in \mathbb{N}$ being a natural number that we set, meaning each agent involved in them answers whether he accepts or declines one or more of the offers he has received.

In the episode, after we have checked that a terminal condition has not been met, we pick an agent at random, using softmax with temperature scaling (4.1), so that the each agent has an equal probability to be chosen as the proposer, as long as, he has not been a proposer before. Then, the proposer is allowed to make k proposals, where k is the denoted proposal limit, as mentioned earlier. In turn, each agent receiving a proposal in the form of a coalition formation C and a task T_C for said coalition to complete, runs the same DRL algorithm as the proposer, in order

to decide how to respond to the proposal.

The proposer, in order to make k proposals, uses ϵ-greedy policy. Where in ϵ-greedy, *epsilon* refers to the probability of choosing a random action, thus exploring, and with probability 1-ϵ to exploit, choosing the action to get the most utility, according to the agent's action-value estimates.

In our case, if the proposer has to make a random action as a proposal, it has to suggest a random coalition formation C and a random task T_C. For the coalition C it picks a random combination of agent types t_C of any length between 2 and the total number of types, denoted as $nTypes$. Then, from the set of agents A, picks t_C amount of agents that collectively belong to the types of the random combination of agent types. Lastly, a task T_C from the list of tasks T is chosen as the task of the random action that will be proposed to the involved agents.

For that purpose, the responder, first, calculates the amount of proposals it has the capacity to accept, let's say n. Secondly, uses the DRL algorithm to think n actions. It is of notice, to mention that to avoid recursion problems, we impose a condition, that during the response stage the agent assumes that for the actions it considers, all involved agents would agree [1]. Then, by comparing the proposal received, with the n actions suggested by the algorithm, responds to the proposer about accepting or declining the proposal.

In the case, where the proposal is rejected, the proposer has to make another one in its place and the aforementioned sequence unfolds once more. On the other hand, if it is accepted, the proposed coalition C is formed and attempts to complete the proposed task T_C. As it has been stated previously, for a task T_C to be successfully completed, the coalition C has to have gathered a resources threshold. If that threshold, is not met by the coalition C attempting task T_C then a penalty, will be distributed equally among the participating agents, for failing to meet the task's quota. In the case, where the participating agents, manage to meet or surpass that threshold, they are rewarded equally, for their successful endeavor.

In our collaborative environment, agents must strategically form coalitions under

[1] instead of said approach, agents could use any method of choice to estimate the probability of reaching a specific coalitional agreement, i.e., as mentioned in [19]

conditions of structural uncertainty. This uncertainty arises from the ambiguity surrounding the value of synergies among agents.

To model these synergies, we drew inspiration from the Relational Rules (RR) a concept introduced in [35], which dictate how the collaboration value of a coalition is determined by the types of participating agents. These rules are able to encapsulate the dynamics of cooperation, accounting for type contributions within a coalition.

With that in mind, we want our reward function to take advantage of this cooperative structure, evaluating the cooperation between pairs of agent types. The reward function should provide feedback that incentivizes agents agents to form task-completing coalitions. The *reward function* that will be used is expressed in the following manner:

$$R(\vec{t_C}, T_C) = \begin{cases} \sum_{i=1}^{z} \sum_{j=i+1}^{z} m(t_i, t_j) \cdot DoC_{ij} & \text{if } T_C \text{ quota is met} \\ -\sum_{i=1}^{z} \sum_{j=i+1}^{z} m(t_i, t_j) \cdot DoC_{ij} & \text{otherwise} \end{cases} \tag{4.3}$$

Where z, is the total number of types that participate in the coalition C, t_i refers to *type* i and accordingly the same applies for t_j, and $m(t_i, t_j)$ is the function that returns the reward for completing task T_C by a coalition C of types t_i and t_j and DoC_{ij} refers to the degree of cooperation between types t_i and t_j.

Regarding the function m, the way it works is based on the rationale that we explained earlier regarding how a reward for a task is generated. For each task we create a matrix of $1 \times g$ dimensions where g is the number of all possible combinations of distinct types of agents in a coalition, which means the total number of all pairs of types. Then by employing a Gaussian distribution we fill the aforementioned matrix with integer values that represent the reward to be given to a pair of types in case they participate in the completion of the task.

By summing over all pairs of types in (4.3) we calculate the reward for a coalition C that completes the task T_C based on the type vector $\vec{t_C}$, in that way the reward that is generated reflects the synergy between the types of agents in accomplishing task T_C.

In case the quota of the task T_C is not met, as we have explained earlier, the reward is turned into penalty as shown in the second branch of (4.3.)

Then depending on how and when we choose to, the degrees of cooperation amongst the agent types are updated based on the experiences of the agents participating in coalition formation during the episode. At the end of a round the coalitions formed are maintained and we proceed to pick another agent as the proposer using softmax distribution with temperature scaling (Eq. 4.1) and another turn begins, like we explained, until a terminal condition is met.

For the update of the *Degrees of Cooperation* (*DoC*), we either use a simple update protocol, or a GNN-based one. The simple update protocol is based on wanting each agent to update the degrees of cooperation of its type with the other types as it cooperates with them in coalitions.

Simple Update of DoC An important aspect of the "simple update" method is to effectively balance positive and negative updates to the cooperation values. Agents should consider not only their own interests, but also the potential for future cooperation and the overall system performance.

To that end, we update the degrees of cooperation between the types that agent i has participated within in a coalition C, from our agent i's point of view:

$$t(C,i)_{xy} = t(C,i)_{xy} + eval(C,R) \cdot t(C,i)_{xy} \tag{4.4}$$

Here the term $t(C,i)_{xy}$ represents the cooperation value between the *types* x, y from the perspective of agent i from participating in coalition C and $eval(C,R)$ refers to a function $eval()$ that takes as input the reward R that was distributed to the agents of a coalition C and outputs a modifier which is then multiplied with the previous value $t(C,i)_{xy}$. This works as a scaling factor for the update of the cooperation values, to control the magnitude of the cooperation value updates, allowing for gradual changes rather than abrupt spikes.

The $eval(C,R)$ function helps the agent evaluate the outcome of the formed coalition in order to later adjust the cooperation value between types x, y. To explain

further, the agent compares the reward R it receives from forming the coalition C with the rewards it has received from the other coalitions it has formed up to this point.

$$eval(C, R) = \begin{cases} modifier & \text{if R} > \text{best\_payoff} \\ 0 & \text{if R} = \text{best\_payoff} \\ -modifier & \text{if R} < \text{best\_payoff} \end{cases} \tag{4.5}$$

If it finds the newly formed coalition performed better than other coalitions, then the agent will increase the co-operational value between its type and the other types participating in the coalition. Else, if the reward is the same, the cooperation values will remain unchanged, and lastly, if the reward is worse or it receives a penalty for forming that coalition, in case the coalition is unable to complete successfully the task it tried, the agent will decrease the co-operational values.

The amount by which an agent will update the cooperation value with the types in the coalition would be proportional to the reward received. Meaning that, after comparing the achieved reward with that of previous coalitions, we find the percentage(%) of how close the reward is to the best received reward so far, represented by $modifier$ in the above equation.

For example, assume that an agent joins a coalition C_1 and receives reward R_1, then it joins a coalition C_2 and receives reward R_2. Comparing R_2 with R_1 it finds out that R_2 is 25% more than R_1, thus more beneficial, so it decides to increase the cooperation value with the other types in the coalition by 25% of the previous cooperation value.

The above formula can also be used from the responders after they have accepted some proposed coalition formation, since they will have a change in their remaining resources as well as acquired some payoff depending on the outcome of the task completion.

For the final update of the degrees of cooperation, after all our agents have formed coalitions, we average out the updated values for the degrees of cooperation that our agents have learned, based on the chosen actions and the resulting rewards

they have experienced throughout the episode.

For example, the update of the cooperation between *types* B, C would be expressed in the following manner:

$$\widetilde{t}_{BC} = \frac{\sum_i^N \widetilde{t}_{BC}}{|N|} \tag{4.6}$$

Where N is the total set of agents on our setting. The term $\frac{\sum_i^N \widetilde{t}_{BC}}{|N|}$ refers summing the adjustments that each agent of type B has taken note of type C by considering the coalition outcomes where both types participated. Each agent involved in coalitions with types i and j, use those $\widetilde{t}_{ij}$ values to be its DoC_{ij} estimates of the types i's and j's degree of cooperation.

A figure representing the outline of the simple update is shown below:

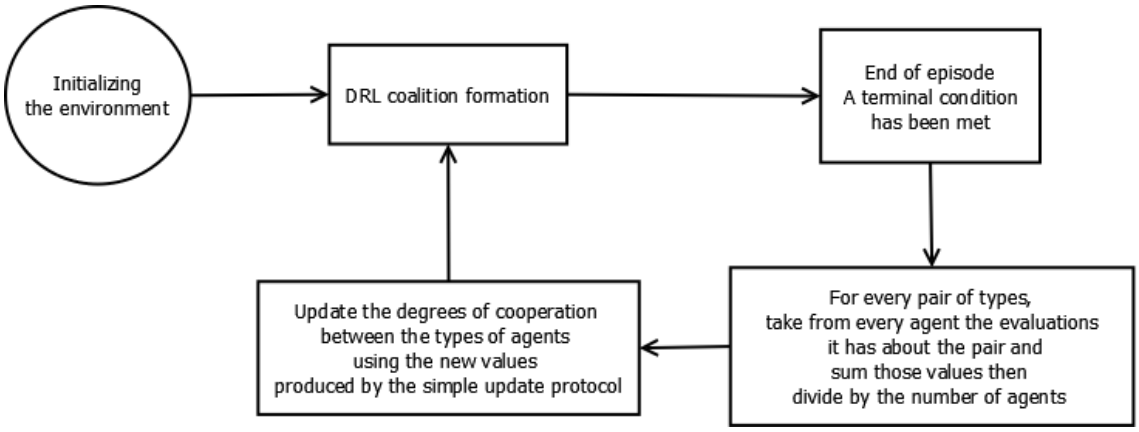

Figure 4.1: Simple update protocol

GAT update of DoC The other method, that will be used as an update mechanism of our system, is using a GAT [37] for the update of the degrees of cooperation between the types. Evidently, taking the mean of a cooperation between two types as the means of updating its value will not be able to cope with large settings, i.e. will be unable to scale with the number of agent types. For such a case, a GAT could be implemented to help us with a more precise way of updating the degrees of cooperation between the agents.

After our agents have formed their coalitions during the DRL phase, we can represent with a graph the environment after the Sequential Overlapping Coalition Formation stage, by having the agent types as nodes, with the node features being the number of agents of a type and the reward they have collected. The edges of the graph would represent the connections between the types, essentially, the synergies between them. Then, GAT comes into play, as with said graph as input it will in turn calculate the attention coefficients for the graph. Those attention coefficients, represent how important the node features of a node are for the others. Thus, updating that way, with the help of a powerfull GNN such as GAT, the cooperation values between the agent types can be updated.

The updated values will then be fed to the environment before the sequential overlapping coalition formation process is repeated.

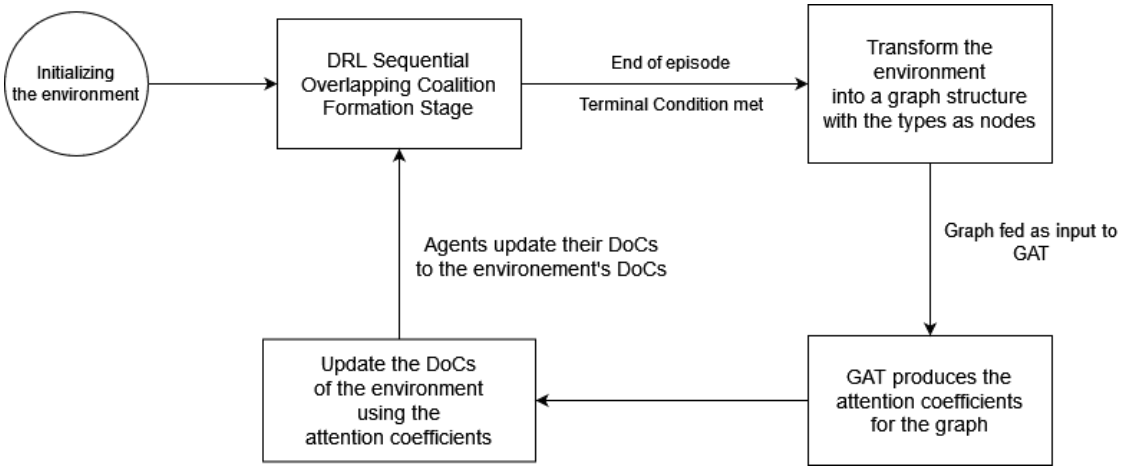

Figure 4.2: GAT update protocol

Before we move on to discuss our DRL algorithm, it is important to note some-

thing regarding the methods of updating the DoC. One could use approaches where such update protocols (either *simple* or GAT) are not globally communicated, but instead agents use their own updates (i.e, running their own GAT in the case of the GAT update protocol).

4.2.2 Method of update of DoC values: Details and Complexity analysis

Moving on, we elaborate on the two methods that we are going to use in order to update the degrees of cooperation between the types of agents. Those methods being the simple protocol and the GNN based protocol that we have explained briefly in Section 4.2 .

Although they both execute the same functionality, they follow inherently different approaches in doing so. To make their differences more apparent, we will now present in detail how in each case the degrees of cooperation (DoC) between two types of agents are updated.

In the first case, we use the simple update protocol presented in 4.1(Eq. 4.6). For argument's sake, we will consider that the update occurs at the end of the episode, meaning that coalitions can no longer be formed due to a terminal condition being met. Now we select a pair of types, for which we will then sum the value that each agent holds for the cooperation of those two types, based from his experiences, and then we divide that sum to the number of agents in our set.

This operation, if we lets say have k number of types, needs to be done $k \cdot (k-1)/2$ times, with the equation being derived from the fact that if we put the types in a $k \times k$ matrix we would want the strictly upper/lower matrix (doesn't matter which since the value for the pair ij is the same for ji). So, for the complexity of the above update protocol where we have integer $\mathbf{N}$ number of agents in our set and integer k types of agents, we would have $N \cdot k(k-1)/2$ operations. The complexity for our simple update protocol can be expressed as $\mathcal{O}(n^3)$. This arises from the nested loops we have: one loop iterating over the types of agents k, and an inner loop performing operations that are quadratic in terms of k, specifically $k \cdot (k-1)/2$ times.

In the case of using a GAT for the update of the degrees of cooperation between types of agents the route we take is different. Again the update happens at the end of an episode, however instead of summing the values that each agent has for the synergy of a pair of types, we transform our environment into a graph structure with the nodes representing the agent types, having as node features the agents belonging to the type that the node represents and the total reward they cumulated. The edges of said graph symbolize the synergies between the agent types, and we feed it to the GAT as input.

The GAT in turn will produce the attention coefficients, which represent the updated degrees of cooperation DoC between the agent types, based on what transpired during the episode, as shown in (4.2). It is important to note, that instead of taking every agent's view of the DoC values for a pair of types and executing the aforementioned operations in the simple protocol, we pass through the Graph Neural network that is GAT, a more well-rounded view of the outcome of an episode, having it produce the update values through the attention coefficients mechanism, which represents the significance of a node's features to it's neighbors.

A benefit of using a GAT is that we have to calculate once the attention coefficients for the graph we give as input and we are able to update all the degrees of cooperation between the types of agents, whereas in the simple update we would have to take the values of all the agents for a certain pair of types to update it and repeat that process for every possible pair of types.

In this GAT-based update protocol (4.2), we need to take into consideration the complexity of both constructing the graph structure as well as applying the GAT to it.

Let's start with the graph construction. For N agents and k types, we create a graph with k nodes (each representing a type) and the node features include the number of agents belonging to that type and the total reward they have accumulated. Since there are k types, the construction of the graph involves $\mathcal{O}(k)$ operations.

The constructed graph is then passed through the GAT, which involves processing each node and its connections. As it has been stated earlier, each node has two features, leading to $\mathcal{O}(2k) = \mathcal{O}(k)$ operations per node. Having k nodes, the overall

complexity of passing through the GAT is $\mathcal{O}(k^2)$ due to the fully connected nature of the graph.

Afterwards, the GAT produces the attention coefficients for each node, representing the updated degrees of cooperation DoC between agent types. This step involves $\mathcal{O}(k)$ operations, as each node's attention coefficients are computed independently.

Comparing this to the simple update protocol, the GAT method has a computational complexity of $\mathcal{O}(k^2)$ due to the graph structure. This means that as the number of agent types k increases, the computational complexity of the GAT method grows quadratically.

To give an example let's say we have 20 agents and 5 types. Let's start with laying some foundations for the protocols, beginning with the simple update protocol. With 5 agent types, we have $5 \cdot (5-1)/2 = 10$ unique pair of types. For each pair, we calculate the sum of values that each agent holds for the cooperation of those two types based on their experiences. Then, we divide that sum by the number of agents in our set (for our example, 20). This operation is performed for all 10 pairs.

Now, using the GAT protocol, we transform the environment into a graph structure where nodes represent agent types. Node features include the number of agents belonging to the type and the total reward they have gathered. All node are connected since each type can potentially cooperate with every other type. We pass this graph structure as input to the GAT. The GAT produces attention coefficients, representing the updated degrees of cooperation (DoC) between the agent types based on the episode's outcomes. These updated DoC values are obtained in one pass through the GAT (see 4.2). The example's graph structure given to the GAT is as follows:

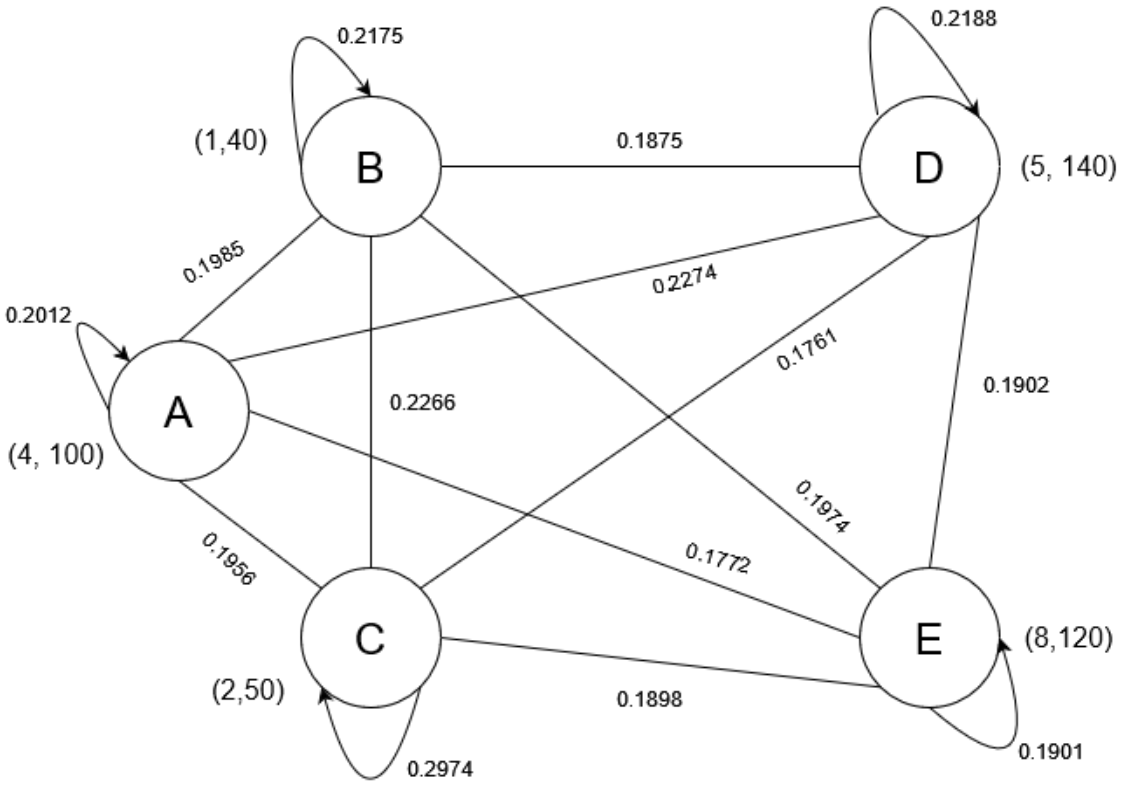

Figure 4.3: Example GAT

Now, in the example scenario, say we use the simple protocol for the pair of types A and B and the sum of values based on agent experiences is 15. Dividing it by the total number of agents (20), we get an updated *DoC* values for that pair of 0,75.

Using the *GAT* protocol, the graph structure with nodes representing types A, B, C, D, and E is created. Each node's features include the number of agents belonging to that type and their total reward. For instance, if Type A has 4 agents with a total reward of 100, its node features would be (4, 100). This graph structure with the described nodes and node features is then used as input to the *GAT*.

The *GAT* processes this graph and produces attention coefficients through the message passing procedure we have described in Chapter 2 (Section 2.3.2). These coefficients reflect the updated *DoC* values between the types. For the above example, as shown in 4.3, the *DoC* between A and B produced by the *GAT* is 0.2776.

The key distinction is that with the *GAT* protocol (4.2), we process the entire

graph once to obtain the updated DoC values for all pairs of types, whereas with simple protocol, we perform calculations for each pair separately.

DQN for Sequential Overlapping Coalition Formation

As shown in the above layout of our environment's setting (See Algorithm 3), we use a DRL algorithm both in the case of the proposer, as well as, in the stage of responding to proposal by the involved agents. The algorithm we decided to implement in our case, was none other than DQN algorithm, which we considered to be an ample candidate to use in our setting. To that end, however, we needed to make some modifications compared to the standard form of the algorithm (See Algorithm 2). The main challenge that we faced that lead to the modified version, is the fact, that the action space of our sequential overlapping coalition formation environment is extremely large, since an agent can propose for coalition formation any length between 2 and $nTypes$, and any task from the list of tasks T.

To provide an example with some numbers, if we have $nTypes = 9$ and total number of tasks is 18, the total number of combinations of any length between 2 and 9 is 48.610 and if we have it multiplied by the number of tasks, we get 874.980 possible actions, making our action space very hard to cope with standard DQN.

To tackle the above problem, we would have to find a way to translate a random natural number from 0 to 874.980 to a proposal, and then how that would be applied during training and to the environment. Also, it would be required to change how both the state and actions are represented, alongside carefully constructing the training function for the DQN.

Below, we showcase the DQN algorithm for Sequential Overlapping Coalition Formation:

Algorithm 4 Deep Q-Network (DQN) for Sequential Overlapping Coalition Formation (SOCF)

Require: Initialize replay memory D with capacity N
Initialize action-value function Q network with random weights θ
Initialize target action-value function $\hat{Q}$ network with weights $\theta^- = \theta$
Set exploration rate ϵ, ϵ_{min}, γ and α
Set replay batch size N_{batch} and update frequency u_f
for each episode **do**
Reset the environment
Agent is chosen at random as the proposer to make k proposals
while n=1, k **do**
Proposer chooses action a_t using ϵ-greedy policy
Responding agents run DQN to respond
if proposed action a_t is rejected **then**
Proposer makes another proposal
else
Execute action a_t, observe reward r_t and next state s_{t+1}
Store transition $(s_t, a_t, r_t, s_{t+1}, done)$ in D
Sample N_{batch} random transitions $(s_i, a_i, r_i, s_{i+1}, done_i)$ from D
Calculate Q-values for states and $next\_states$ with Q network

$$Set\ y_i = \begin{cases} r_i & \text{if episode terminates at step i+1} \\ r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \theta^-) & \text{otherwise} \end{cases}$$

Update action-value function by minimizing the Huber loss:

$$L_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & if\ |(y - \hat{y})| < \delta \\ \delta((y - \hat{y}) - \frac{1}{2}\delta) & otherwise \end{cases}$$

Update target $\hat{Q}$ network every u_f steps: $\theta^- \leftarrow \theta$
Update the DoC between the types of agents

At first, we have to initialize the replay buffer memory D with capacity N, which will be used to implement the experience replay functionality that is present in the original algorithm, to train DQN. We will use it to store experiences from the environment, which in our case, will be information from the coalition formation that unfolds in the environment (See Algorithm 3).

Next, is the initialization of both the policy Q-network and target Q-network with random weights θ and θ^{-} respectively, which are the neural networks we use, for calculating the Q-values for each state-action pair. In our case, the state representation for an agent would be, the amount of resources r it has left, the agent's index, and the coalitions it has formed, where for the action representation it would be the coalition that is proposed to be formed and the task that accompanies it.

After that, we set the values for exploration rate ϵ and minimum exploration rate ϵ_{min}, that are vital in using ϵ-greedy policy for action selection, with the ϵ_{min} being the threshold for the decay of the ϵ as the DQN transitions from exploration to exploitation to stop, so that the agent will explore with ϵ_{min} probability. Then, the N_{batch} represents how many experiences the DQN will pull from memory D when it trains, and the u_f is how frequent should the target Q-network be updated. Before the DRL-SOCF stage begins, we set the values for the discount factor γ that is a constant between 0 and 1 used to ensure that the sum converges. A lower γ makes rewards from the far future less important than the ones in the near future for our agent. It also encourages agents to collect rewards closer in time than equivalent rewards that are temporally far away in the future [42]. Last but not least, is the setting of the learning rate α which regulates the weights of our neural network concerning the loss gradient, meaning how much we accept the new value vs the old one.

Now, for what transpires during an episode using DQN in our setting, firstly, a proposer is picked in the same manner as we have discussed earlier on, using softmax with temperature scaling (4.1). Then, using ϵ-greedy policy, the policy network chooses an action a_t for the proposer to suggest. As mentioned, the action space for our DQN is quite large, to work around that problem and translate a random number into an action, we did the following. If we divide the random natural number with the total number of tasks, we get the index of the type combination and the

mod is the index from the list of tasks. To give an example using the numbers we mentioned earlier, $nTypes = 9$ and total number of tasks is 18, the total number of combinations is 48.610 and multiplied by the number of tasks, we get 874.980 possible actions. Picking a natural number, between 0 and 874,980, let's say 45,687, we divide it with 18, $45,687 \div 18 = 2,538$ with remainder 3, which means that 45,687, refers to the types combination with index 2.538 and the task at index position 3 of the list of tasks T.

After the proposal is made, each agent involved has to respond about accepting/rejecting the proposal. To reach that decision it also uses DQN, though it does not run it like the proposer. Instead, the responders use the DQN to facilitate a similar logic to what we mentioned earlier (See Algorithm 3). The responder calculates how many proposals is able to accept, n; then uses the DQN to find the n best actions based on its current state; and then compares those actions with the proposal it received. If the proposal happens to be one of those n actions, or if, the coalition has a similar setup of types to one of the n actions, like we discussed earlier on the section (Eq. 4.2), it accepts the proposal. Otherwise, the proposal is rejected and the proposer has to make another one.

Reaching an acceptable proposal a_t, it is executed on the environment, yielding a reward/penalty for the involved party. The generation of said payoff r_t, is the same as stated earlier in Eq. 4.3. Then, we can represent the next state s_{t+1}, as well as get informed from the environment on whether we reached a terminal state or not through the *done* variable. Next, we store the transition $(s_t, a_t, r_t, s_{t+1}, done)$ to memory D since we have all the needed components.

Now, if we have enough transitions in memory, meaning, at least N_{batch}, we can train our DQN with experience replay. We start by sampling N_{batch} experiences from memory D, then we calculate the Q-values for the states s_t sampled and do the same for the next states sampled s_{t+1}.

Afterwards, we calculate the Q-values using the target network for the actions a_t with the use of Bellman equation: $y_i = r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \theta^-)$.

Subsequently, we want to update the policy network to minimize the loss between the expected and the true Q-values. To that end, we use the Huber loss [3], to minimize that difference. The important aspect of Huber loss, that makes it suitable

for our case, is that it acts like the mean squared error when the error is small, but like the mean absolute error (MSE) when the error is large– this makes it more robust to outliers when the estimates of Q are very noisy [42].

Thereafter, if we have completed u_f steps, we update the target network with the trained weights of the policy network, (i.e., set $\theta^- \leftarrow \theta$) and decrement the ϵ parameter to gradually balance exploration and exploitation. Finally, we update the DoC between agent types, informing the agents of the update cooperation values between the types, either by using a simple update protocol (4.1), or a GNN based one (4.2), as also mentioned earlier (See Algorithm 3).

Chapter 5

Experimental Evaluation

In this section, we will first describe our experimental setting and setup in detail, and then present the experimental evaluation of our approach.

5.1 Custom Gym Environment Setup

To evaluate the performance of the proposed algorithms in the context of Sequential Overlapping Coalition Formation ($SOCF$) with Deep Reinforcement Learning (DRL), it was imperative to create a tailored environment that faithfully emulates the intricacies of the problem.

The environment, formulated as a DRL problem, was designed to simulate a scenario involving multiple agents, each belonging to specific types, and a set of tasks. This simulation was achieved through OpenAI Gym, a widely used toolkit for developing and comparing DRL algorithms [27].

The observation space contains the state, action, reward, next state and done variables that we have expained earlier about their representation and usage in the environment (See Algorithm 4).

The action space, contains all the possible combinations of the agent types alongside the tasks, since as clarified before, a proposal specifies a coalition formation containing specific agent types, and a task to complete.

Upon initialization, the environment is provided with the list of agents, the tasks, the agent types and their possible combinations, as well as, the degrees of

cooperation (DoC).

The reset method is responsible for resetting the environment to its initial state at the commencement of each episode. It accepts the lists of agents, tasks, and initial agent configurations. Resources, coalition formations, proposal counts, rewards, and coalition payoffs are reset for each agent. Additionally, every task's completion status is set to false.

The step method, being the most crucial part of the custom environment, is responsible for executing the chosen action (coalition and task) by the proposer agent. It calculates the resulting reward and updates the environment accordingly. The coalition is added to the list of formed coalitions of the participating agents, and the reward is determined based on the coalition's performance in the task. The achievable status of the task is checked, and appropriate rewards and penalties are applied to the coalition members. The proposer's epsilon (exploration rate) is reduced, and the proposal counter is incremented. The method returns the reward and a boolean indicating whether the episode is complete.

Finally, we added another method, for assessing whether the environment has reached a terminal condition (See Algorithm 3), signifying the end of an episode. Notifying us that way, about the termination of the episode, so that we can then ensue the update of DoC before the environment resets for the next episode to start.

This custom Gym environment, serves as the foundation for our experiments, providing a structure for the evaluation of our algorithms and update protocols. It encompasses fundamental elements such as tasks, agents, types and mechanisms for task completion and reward distribution. The observation and action spaces are customized to capture relevant information for the learning process. The methods of the environment aid the execution of tasks, the calculation of rewards and determination of terminal conditions, enabling the agents to interact and learn within this setting.

5.2 Experimental Setups

Before commencing the experiments using the methods and temporal alignments, we have discussed earlier, we should make some parameters adjustments so that it

is easier to understand the setup of the environment during each experiment.

As we have stated, it is our intention to test our models using both a small set of agents and a larger one. For the Setup A, as we will refer to it from now on, we have:

- Agents 50
- Number of types 6
- Number of tasks 12
- Proposal limit 5

Whereas for the Setup B, we will be using the larger values listed below:

- Agents 250
- Number of types 10
- Number of tasks 25
- Proposal limit 10

The reason behind the values of the environment's parameters for our two setups, is to test the scalability of our approach. We also consider Setup B to be quite realistic, and so is Setup A, at least as far as the number of types is concerned (for instance, there are typically 5 credit levels in finance, 5 types of renewable energy sources, about a dozen main and secondary branches in army, and so on).

The state of the environment as perceived by the decision making agent controlling its own MDP, consists of the amount of the agent's remaining resources. This value is crucial for the guiding of the agent's decision-making behavior. The amount of the remaining resources is an indicator of the number of coalitions that the agent has the capacity to participate in the future. To calculate the number of coalitions n_{pc} the agent can form, we divide the resources r, of an agent, with the contribution of the agent's type $t_{contribution}$:

$$n_{pc} = \frac{r}{t_{contribution}} \tag{5.1}$$

For example, we take an agent with the maximum amount of resources, which for the sake of the example is $r = 200$. The agent belongs to a type t which has the minimum type contribution, $t_{contribution} = 10$. Then, our agent in that case, has the capacity to participate in up to 20 coalitions.

Similarly, an action consists of two elements, a coalition C and a task T. The C element is a list of agent types of length between 2 and $nTypes$. Note that, here, an agent type can be included more than once. To calculate the combinations of a certain length l from a set n with replacement (duplicates are allowed) we follow the formula below:

$$Combinations(n, l)_{Replacement} = \frac{(n + l - 1)!}{l!(n - 1)!} \tag{5.2}$$

The other element, T is a task that C is called in to complete. Note that, each task has particular resource requirements in order to be completed successfully. If coalition C meets this quota, participating agents are rewarded with a positive payoff generated by the task depending on the type composition of the coalition (Eq. 4.3). Otherwise it receives a penalty that is equal to the negative value of the payoff that the task would generate.

As concerns Setup A, the size of the action space is the number of all possible combinations of any length between 2 and 6 ($nTypes$) with replacement multiplied by the number of tasks, which in this case is 917 combinations times 12 tasks, resulting to 11,004 possible actions. For the case of Setup B, this equals to 4,618,625, considering that nTypes is 10 (184.745 combinations) and the number of tasks is 25.

As we have described earlier in (Algorithm 3), an action is selected using an ϵ -greedy policy. This means that we choose a random action from the possible ones with a probability ϵ.

In order to come up with a coalition and a task to form an agent action, we first generate a random integer in the range (0, *Total type combinations*$\cdot$ *Total Tasks*-1). Then, we divide this integer with the total number of tasks, and thus acquire the index of a particular type combination that is referred to by the quotient. The remainder of this division (modulo operation) corresponds to the task.

To summarize, the state-action space of our environment depends on the potential of an agent to form coalitions, the number of types, and the number of tasks. The representation of our state-action space can be formulated as below:

$$\text{state-action pairs} = n_{pc} \cdot \text{Total type combinations} \cdot \text{Total tasks} \tag{5.3}$$

Using the numbers we have provided as an example earlier, we can calculate the state-action space for each setup for an agent with $n_{pc} = 20$. For Setup A, the total of state-action pairs is:

$$n_{pc} \cdot \text{Total type combinations} \cdot \text{Total tasks} = 20 \cdot 917 \cdot 12 = 220,080 \tag{5.4}$$

Respectively, for Setup B, we have 92,372,500 state-action pairs.

Another important aspect of our experimental setups is that apart from the DQN algorithm that we have explained earlier (See Algorithm 4), we will also use $Q-learning$ as another baseline to compare our implementation.

The $Q-learning$ we used is a modified version of the standard algorithm (See Algorithm 1), in the essence that instead of having a table with all possible state-action pairs, due to the exremely large state-action space that our environment has, we implemented a dynamic table for our $Q-learning$ to use.

The state represantation for our $Q-learning$ will be the proposer's remaining resources r, whereas the action is represented as the coalition C that it proposes to be formed alongside the task T for it to complete.

For the $Q-function$ that the agent will use during DRL to calculate the Q-value of a state s_t will have the following form:

$$\begin{aligned} Q_i(r_i, C_i, T_i) = \quad & Q_i(r_i, C_i, T_i) \quad + \\ & \alpha \left[r_{t+1} + \gamma \max_{\alpha} Q(r_i', C_i', T_i') - Q(r_i, C_i, T_i) \right] \end{aligned} \tag{5.5}$$

Where r_i, represents the remaining resources that agent i has left. C_i represents the *coalition* that the agent currently proposes to be formed for the completion of a certain task. T_i is the task that the proposed coalition C_i is trying to complete by forming the aforementioned coalition formation.

The α term refers to the learning rate of the algorithm, $\gamma \max_\alpha Q(r'_i, C'_i, T'_i) - Q(r_i, C_i, T_i)$ is the estimation of future rewards, and γ is the discount rate with value between 0 and 1($0 \leq \gamma \leq 1$) that we have mentioned earlier.

From the point of view of the responding agent to a proposition for a coalition formation, it can also use the same Q-function with the appropriate changes in regard to the r_i term that vary according to the agent we examine at each step. However, the proposed coalition as well as the task that it plans to complete remain the same.

5.3 Experimental Scenarios

We chose to experiment on the aforementioned two environmental settings to test the scalability of the various approaches. Our experiments scenarios will be tested in each of the settings, and will be the following:

1. Q-learning with Simple Updates of *DoC* values at the end of an episode
2. Q-learning with Simple Updates of *DoC* values whenever a proposer changes
3. Q-learning with GAT Updates of *DoC* values at the end of an episode
4. Q-learning with GAT Updates of *DoC* values whenever a proposer changes
5. DQN with Simple Updates of *DoC* values at the end of an episode
6. DQN with Simple Updates of *DoC* values whenever a proposer changes
7. DQN with GAT Updates of *DoC* values at the end of an episode
8. DQN with GAT Updates of *DoC* values whenever a proposer changes

In the following subsections we define the meaning of having the update occuring at the end of an episode and at the change of the proposer, as well as, the workings of both the *Simple* and *GAT* updates.

5.3.1 Moment of update of the *DoC* values

Another very important aspect of our coalition formation setting, is the update of the degrees of cooperation between the agent types. As it has been stated earlier, we plan to test the update happening at two different points during the episode.

The first point we will test the execution of the update would be, at the end of the episode. When a terminal condition of the ones stated earlier have occurred and all agents have formed the coalitions they could. Based on their experiences each one has a more informed view about the types of agents it has participated on coalitions with, and that is what its input to the update will be based upon.

Thus at the end of the episode, before we resolve the formed coalitions and reset the resources of our agents etc., we perform the update of the DoC, either by using the simple protocol (4.1) or the GNN one that have been explained earlier (4.2). After that the values of the DoCs between the types are updated from the events that transpired during the episode, and are the values that the agents will have at their disposal at the beginning of the next episode.

The second point where we would test the update is at the end of a proposer's turn, in addition to updating at the end of each episode. In that case, after the proposer has made k proposals and the accepted coalitions have been formed and the reward from those actions has been distributed, the agents involved have acquired some information about the types involved in those formations.

Thus, we can shape the *DoC*s that the agents have at hand, by how each agent was affected by the action that took place during the proposer's turn. That adds to the mechanism of the update a sense of realism, since the trust that a type of agents show towards cooperating with other types, shifts according to how beneficial or disadvantageous each turn is.

Although, in doing so the process of the update as well as the coalition formation, becomes more complicated. That is due to the fact that before choosing the next proposer, we have to perform either one of the update protocols, in order for the *DoC*s to be better informed based on the events of the passing turn, to provide more accurate information to the agents that are going to be involved when a new proposer is picked.

5.4 Experiments

In this section of the chapter, we will show the results of running our algorithms in the combinations and setups mentioned, in terms of duration, reward and *discounted-per-episode total accumulated reward*.

"Discounted-per-episode total accumulated reward" is calculated as follows: we apply a discount rate of δ^i to the reward $r(i)$ accumulated within each Ith episode, and progressively sum these values up to produce the $\sum_i \delta^t r(i)$ function graphs shown in the respective results figures.

This metric was chosen in order to smooth out the reward lines (easing their reading), and also to provide a glimpse at the sequential-across-episodes performance of our (D)RL+beliefs' update methodology. Note however that this metric does not accurately reflect the sequential performance of the agents within each learning episode. Note also that simply using what is usually in RL referred to as discounted total accumulated reward (which discounts rewards at each learning step) is not a straightforward choice in our case, since rewards might be distributed to agents after several steps (decision makers-agents are not guaranteed to form a coalition that receives a reward at the time of their decision-action), and there are also learning episodes involved, with the discount factor used for RL being initialised after the completion of each episode.

Devising a more proper metric to accurately capture sequential performance *simultaneously* within-episodes and across-episodes is left as an open question.

5.4.1 Q-Learning Results

Firstly, we will present our results for the Setup A of our setting, starting with the Q-learning method and for the update methods and their timing combinations we have mentioned previously.

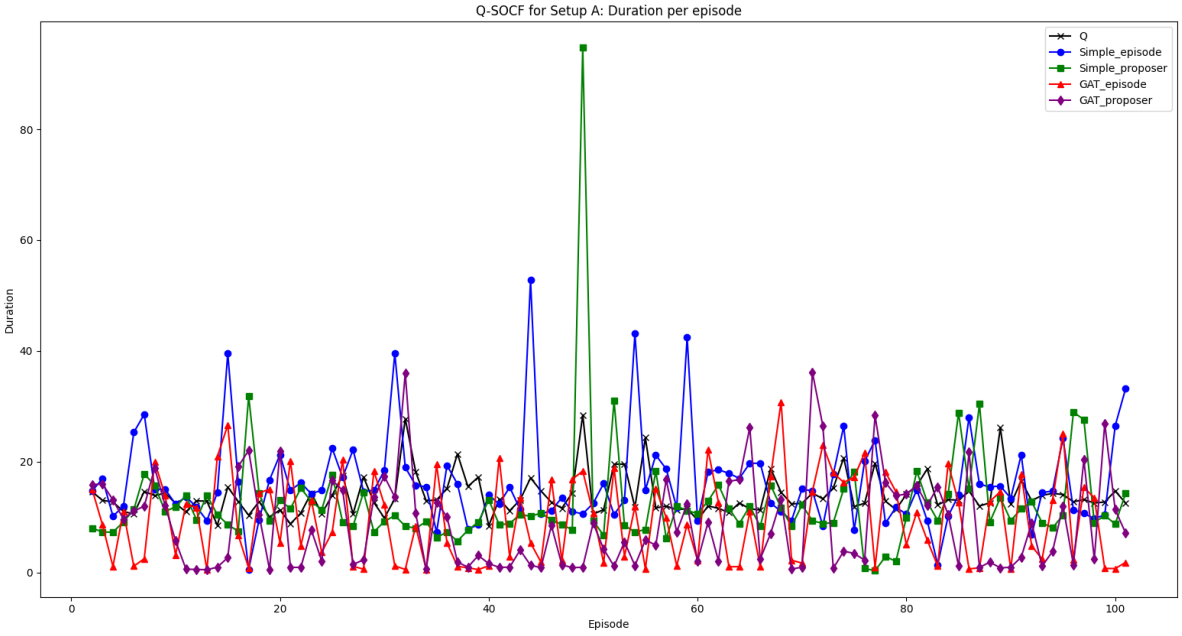

Figure 5.1: Q-learning Sequential Overlapping Coalition Formation, Setup A: Episodes' Duration

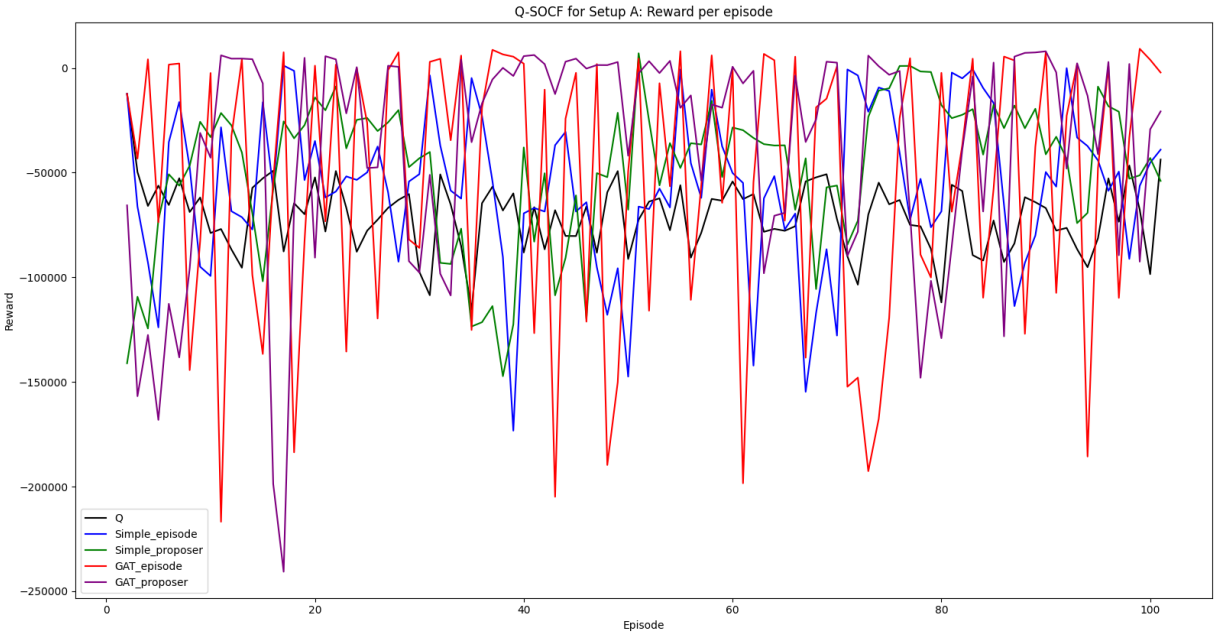

Figure 5.2: Q-learning Sequential Overlapping Coalition Formation, Setup A: Reward per episode

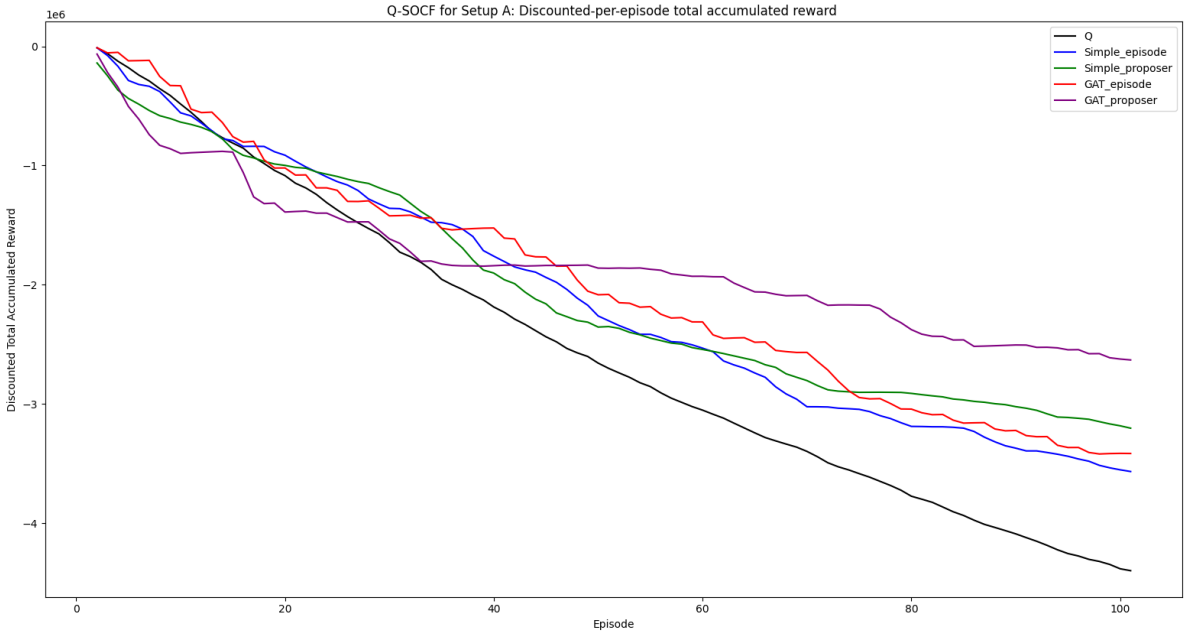

Figure 5.3: Q-learning Sequential Overlapping Coalition Formation, Setup A: Discounted-per-episode total accumulated reward

Table 5.1: Summary Results for Q-SOCF Setup A

Q-learning	**Total Duration**	**Total Reward**
Q Setup A	1,386.385289 sec	-7,097,131.254
Simple_episode	1,626.445527 sec	-5,645,648.442
Simple_proposer	1,234.908739 sec	-4,814,861.056
GAT_Episode	939.4192538 sec	-5,449,631.922
GAT_proposer	870.2227089 sec	-3,818,542.267

As we can see in Table (5.1) the results in terms of reward are not good, although that was to be expected. Since for the coalition formation that is proposed by an agent, it is in the form of a type vector, the total members of a coalition can be at most $nTypes$.

In this small setup, that translates to the coalitions proposed and formed having an observed length between 2 and 6, which depending on the contribution of the types may in most cases not be enough to fullfill a task's quota of resources. This results to, the coalition formation stage being time consuming and not rewarding at the same time.

Next on, we see two figures for the results in terms of duration and reward per episode, regarding the Q-learning algorithm with the simple update protocol occuring at the end of the episode and at the change of the proposer, as well as, the GAT update protocol happening at those two points for Setup B.

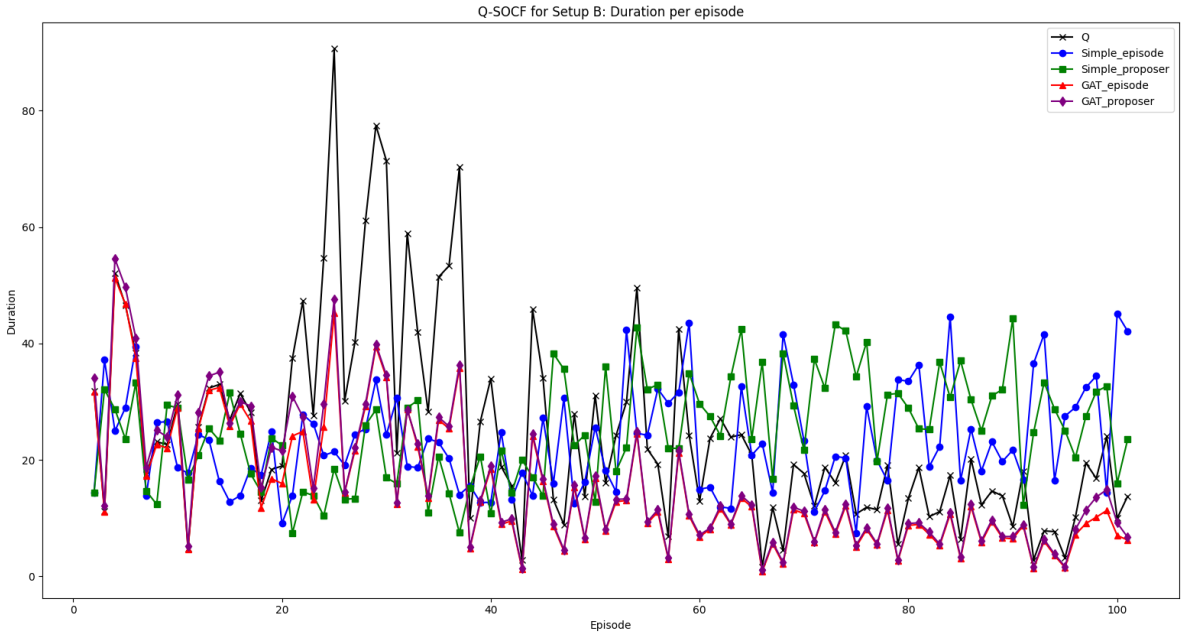

Figure 5.4: Q-learning Sequential Overlapping Coalition Formation, Setup B: Episodes' Duration

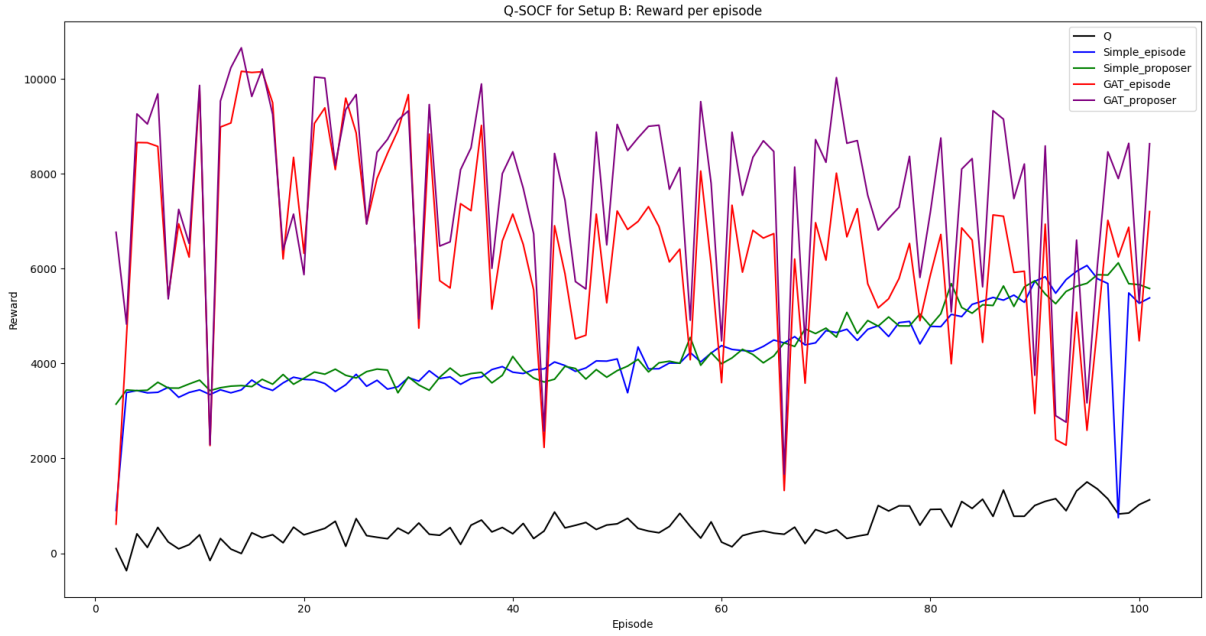

Figure 5.5: Q-learning Sequential Overlapping Coalition Formation, Setup B: Reward per episode

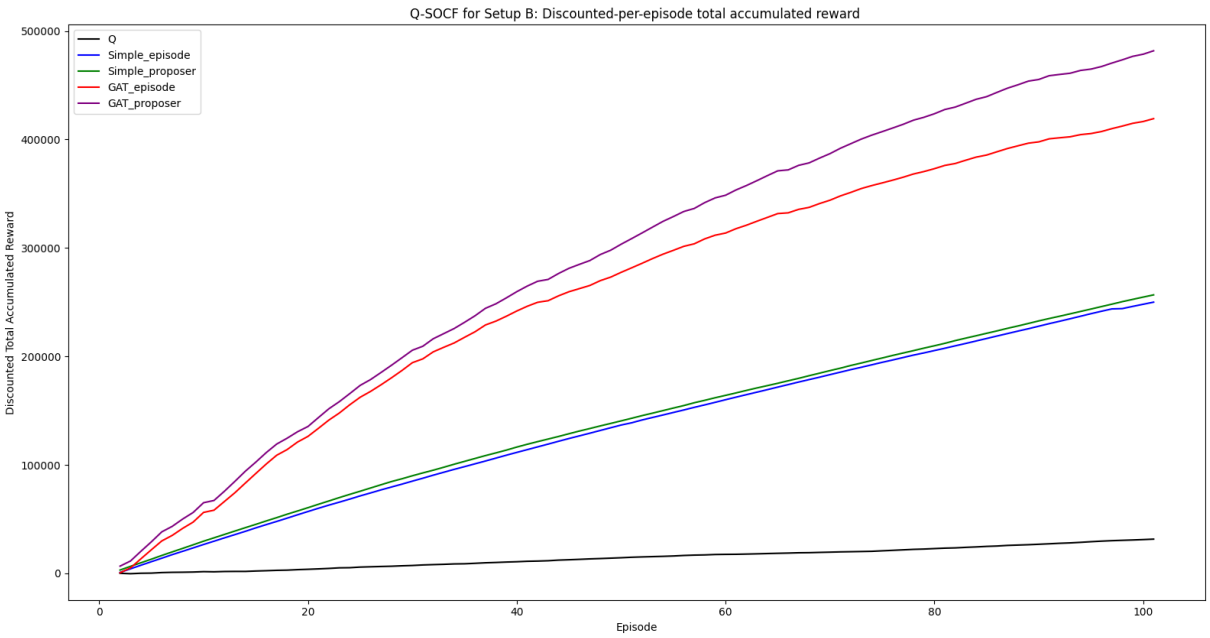

Figure 5.6: Q-learning Sequential Overlapping Coalition Formation, Setup B: Discounted-per-episode total accumulated reward

In the first Figure 5.4, although it is not clearly visible, the *GAT* update method, is significantly faster that the simple one, in both accounts of the different temporal alignment of the update execution. That, occurs because, as we have presented earlier on the chapter, the simple method is more computationally draining ($\mathcal{O}(k^3)$), compared to the *GAT* one ($\mathcal{O}(k^2)$). Also, we can notice, that in both protocols, having the update of the *DoC* occur, at the change of the proposer, makes the running of the environment slower than its counterpart of updating at the end of the episode.

As we can see, in the second Figure 5.5, the *GAT* update protocol, outmatches the simple one, in both the case where the update is at the end of the episode and when it happens at the change of the proposer agent.

This was to be expected, since the use of *GAT* method is more impartial compared to the simple one, since it is based on the data of the environment at the moment it is employed, rather than on the points of view of agents participating on sequential overlapping coalition formation. We could say, that essentially the *GAT* looks at the forest, in order to execute the update of the *DoC* values, whereas

the simple protocol, looks at what each tree views, thus not taking into account as efficiently the bigger picture.

From both those Figures 5.4 5.5, we can reach the following realisations, that the *GAT* protocol is both faster and more rewarding than the simple one and that having the update happen at the change of the proposer agent, making the agents more well-informed about the *DoC* values, it is more rewarding but also a bit more time consuming.

For clarity, we present the total duration and reward per combination below.

Table 5.2: Summary Results for Q-SOCF Setup B

Q-learning	**Total Duration**	**Total Reward**
Q Setup B	2,480.478861 sec	57,299.96232
Simple_episode	2,326.96008 sec	417,585.1183
Simple_proposer	2,517.818551 sec	428,505.3102
GAT_Episode	1,504.226736 sec	645,640.3997
GAT_proposer	1,604.18417 sec	755,879.9084

5.4.2 DQN Results

Subsequently, we will now present our DQN algorithm running the two setup we have presented in the update protocol configurations and alignments that have been discussed.

Starting with Setup A, we present the results in terms of duration and reward per episode, as well as, discounted-per-episode total accumulated reward to provide a clearer picture of their performance.

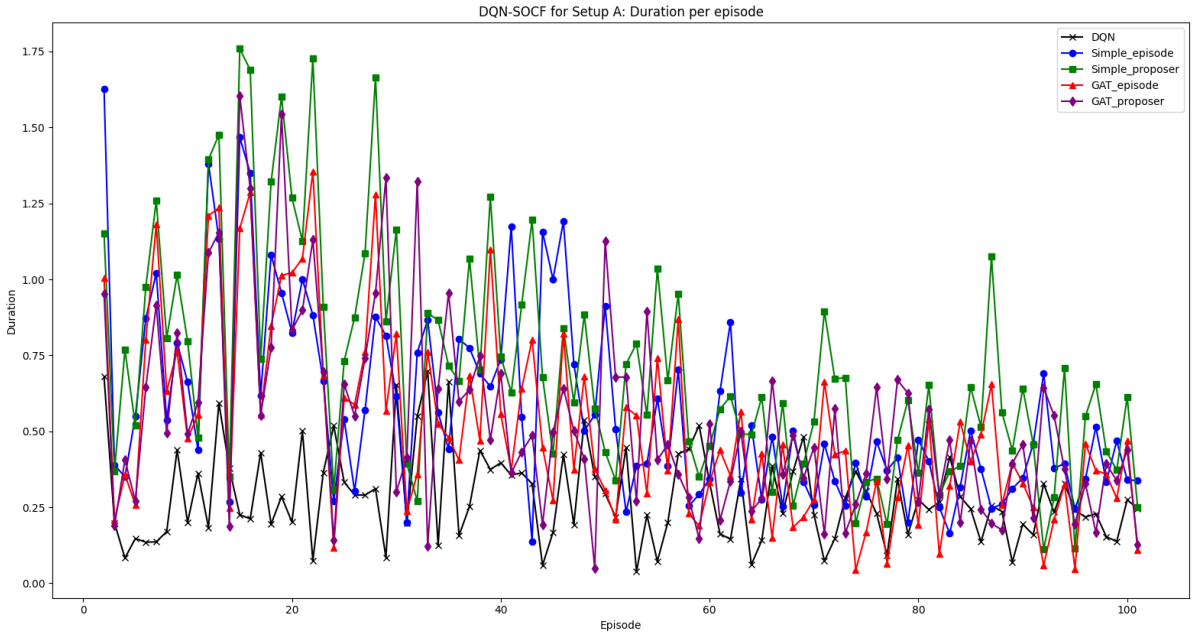

Figure 5.7: DQN Sequential Overlapping Coalition Formation, Setup A: Episodes' Duration

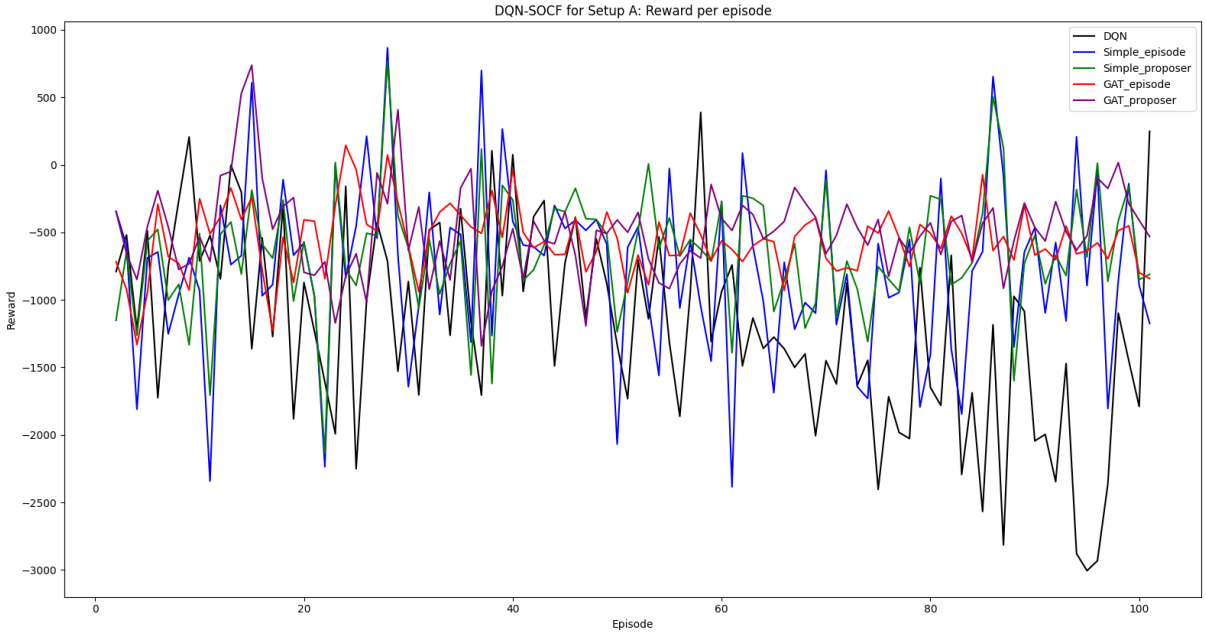

Figure 5.8: DQN Sequential Overlapping Coalition Formation, Setup A: Reward per episode

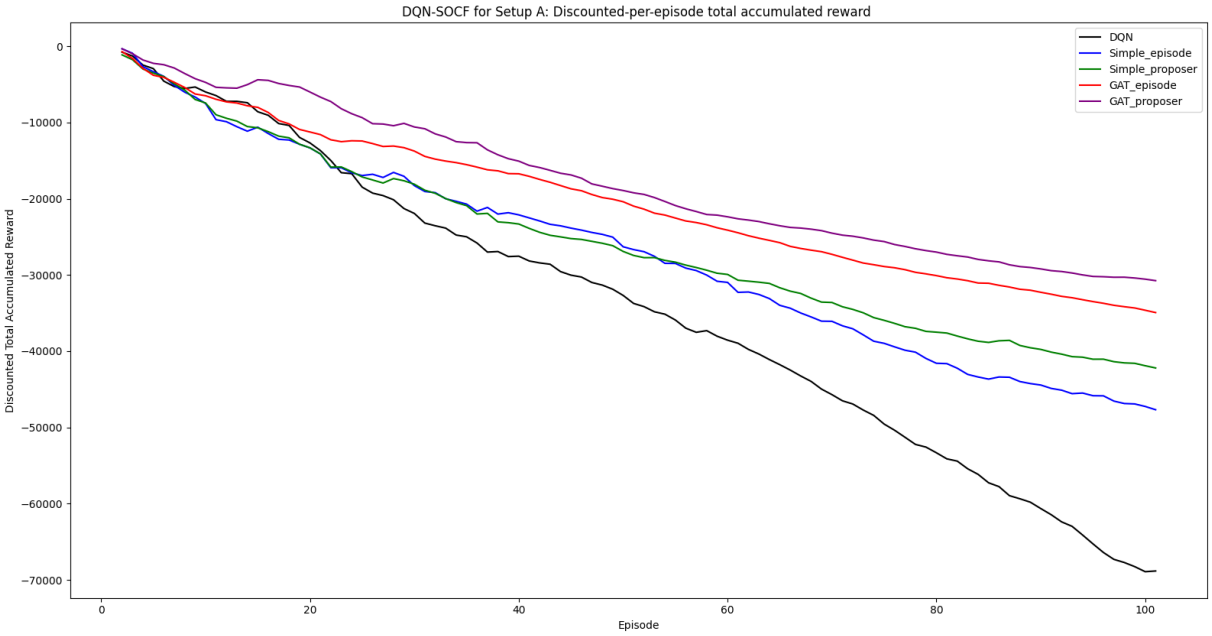

Figure 5.9: DQN Sequential Overlapping Coalition Formation, Setup A: Discounted-per-episode total accumulated reward

Table 5.3: Summary Results for DQN-SOCF Setup A

DQN	**Total Duration**	**Total Reward**
DQN no update	28.22000432 sec	-119,893.9755
Simple_episode	57.08326423 sec	-77,290.44116
Simple_proposer	70.99932337 sec	-65,562.10703
GAT_Episode	50.99985219 sec	-55,794.71404
GAT_proposer	53.39349976 sec	-49,068.47362

Again, we see that the results are not good, for the same reason we discussed in the $Q-learning$ results for the same setup. However, we can see that once more, the use of the GAT protocol, is performing better compared to the $Simple$ one.

An important observation that we make, from the above results, is the benefits, of using any of the two methods to update the DoC, since it showcases a significant increase in the reward as shown in (Table 5.3). Also, again we notice the improvement, that updating when the proposer changes makes. That, shows us that a better informed set of agents is able to make better decisions, whether in making a proposal or answering one.

Next, we test our DQN implementation on Setup B, a significantly scaled up

version of Setup A, making a more challenging case, in the mentioned combinations.

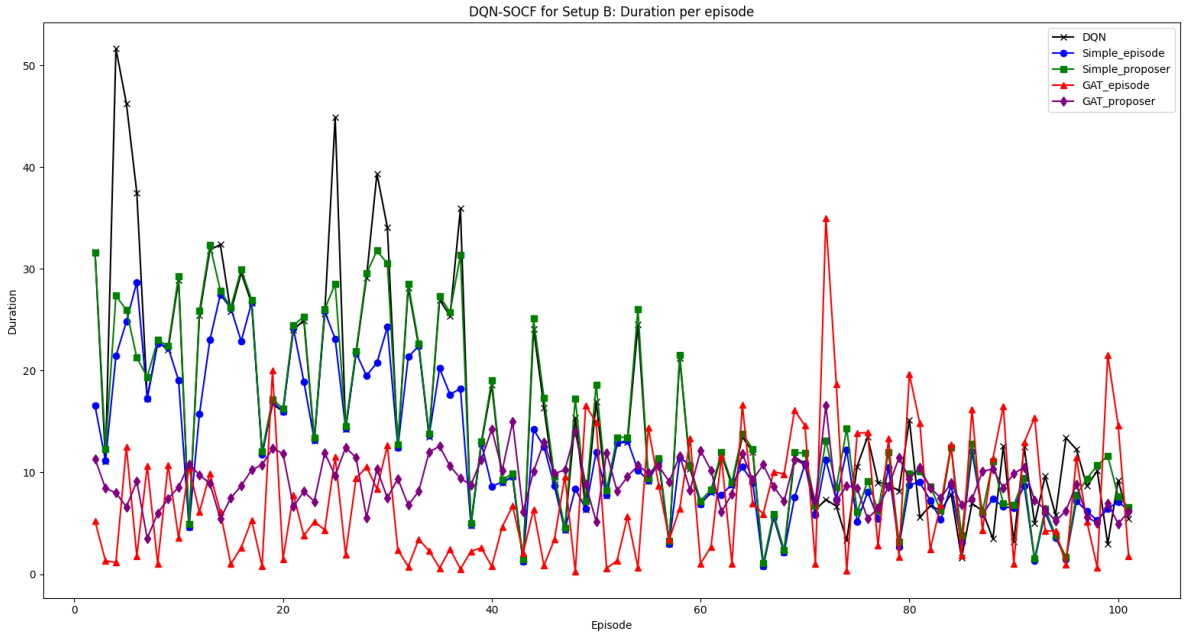

Figure 5.10: DQN Sequential Overlapping Coalition Formation, Setup B: Episodes' Duration

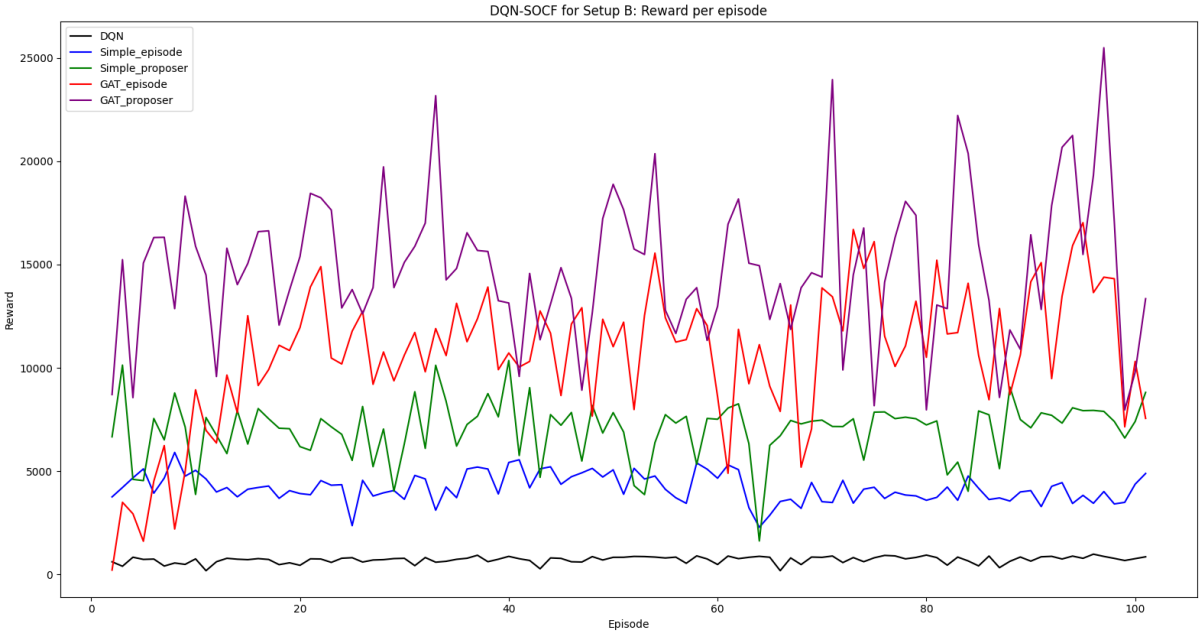

Figure 5.11: DQN Sequential Overlapping Coalition Formation, Setup B: Reward per episode

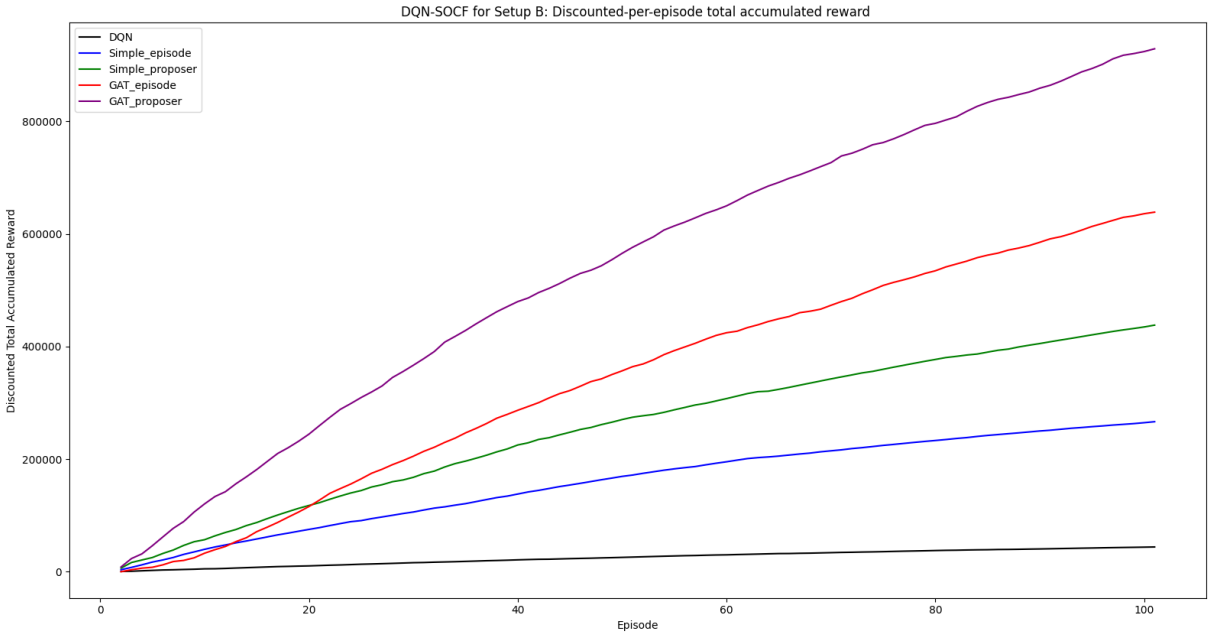

Figure 5.12: DQN Sequential Overlapping Coalition Formation, Discounted-per-episode total accumulated reward

The above figures (5.11 ,5.12) further support the observation we made earlier regarding, the effectiveness of using any of our update protocols for the DoC among agent types. Once again, as it is apparent, the GAT protocol is faster than the simple one, as we see in the above Figure 5.10. The reason why, stands the same as expained in the previous case with the Q-learning variation. However, it is important to note that the use of DQN in the environment makes for better times than the Q-learning one, in all scenarios regarding the time and way of updating the DoC values.

Looking at the second Figure (5.11) that show the reward per episode, for the combinations of update protocol and time of use, we can deduce that again the GAT yields better results. Additionally, the use of DQN appears to speed up the coalition formation process, which is within our expectations, since by its use, the agents have a more informed view of the environment, leading to their better decision-making.

To be able to give a clearer picture of the results in comparison also to the Q-learning implementation, we provide in the below table, the results regarding total duration and reward per combination of our DQN setup.

Table 5.4: Summary Results for DQN-SOCF Setup B

DQN	Total Duration	Total Reward
DQN no update	1,523.357206 sec	71,742.2617
Simple_episode	1,196.521669 sec	419,730.4113
Simple_proposer	1,462.102336 sec	701,510.0778
GAT_Episode	722.992635 sec	1,066,205.425
GAT_proposer	908.72119 sec	1,486,165.278

5.4.3 Q-learning vs DQN

Before we exhibit the comparison between the two algorithms in their best combination respectively, we would like to highlight the difference in their performance without any means of updating the *DoC* among agent types.

Beginning with their perfomace in Setup A:

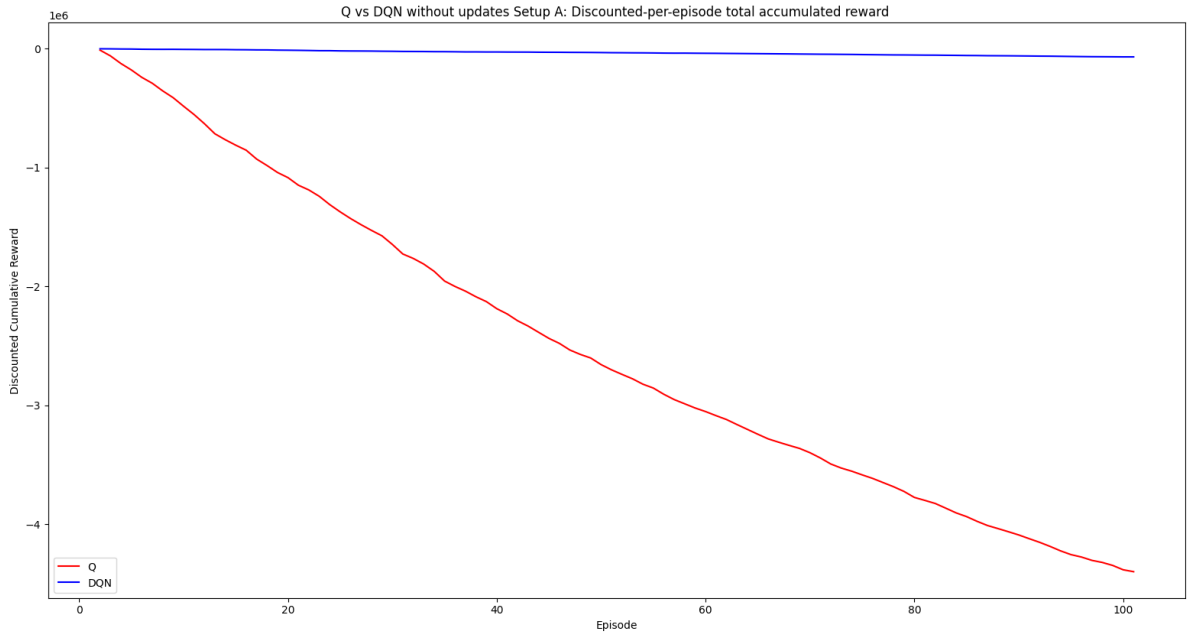

Figure 5.13: Q-learning vs DQN without update of DoC, Setup A: Discounted-per-episode total accumulated reward

Then following in Setup B:

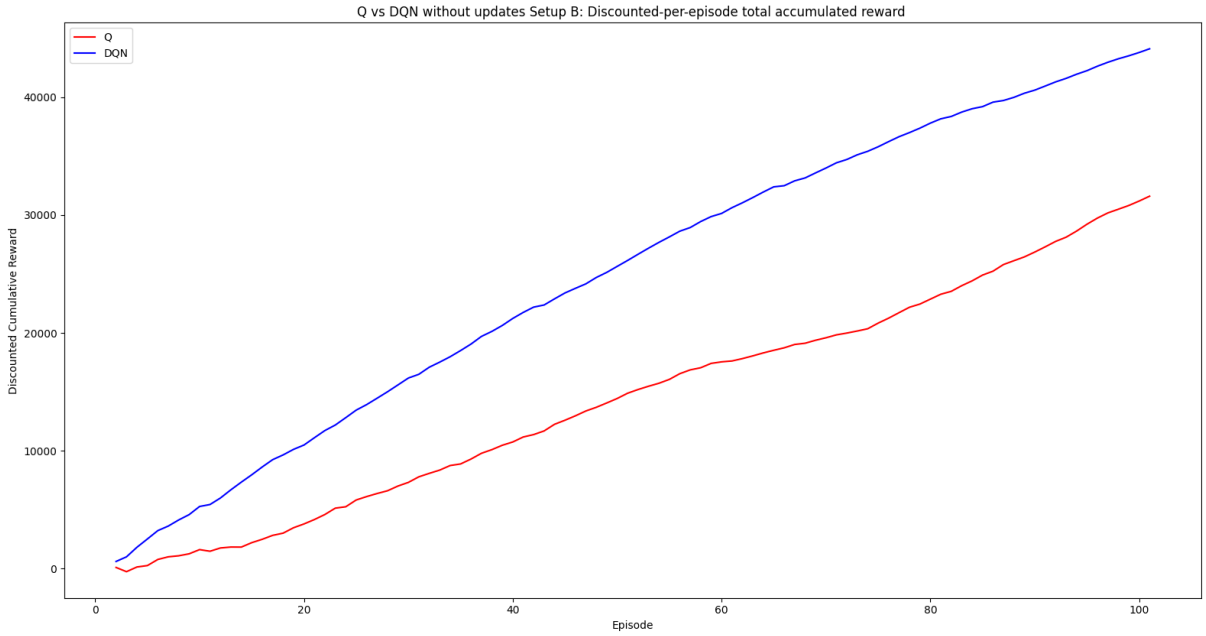

Figure 5.14: Q-learning vs DQN without update of DoC, Setup B: Discounted-per-episode total accumulated reward

What we can deduce from those Figures (5.13, 5.14), is that DQN is better qualified for such a setting. Both algorithms struggle in Setup A, because of the amount of Tasks being too demanding for the set of agents and the reasons we have stated before. However, as it is evident in (5.13), DQN shows better results and a more contained loss in contrast to the $Q-learning$ which performs horribly.

In Setup B, we see both of them having rewarding runs, although it is clear that DQN is more beneficial in that aspect again.

The results for the above figures can be summarized in the table below:

Table 5.5: Summary Results for Q vs DQN without updates

Q vs DQN	**Total Duration**	**Total Reward**
Q Setup A	1,386.385289 sec	-7,097,131.254
DQN Setup A	28.22000432 sec	-119,893.9755
Q Setup B	2,480.478861 sec	57,299.96232
DQN Setup B	1,523.357206 sec	71,742.2617

Lastly, for the sake of completeness, we showcase the best runs of our algorithms, in the prospect of reward, which are in the case of using the GNN-based protocol at the change of the proposer agent, for both our Q and DQN implementations.

First, for Setup A:

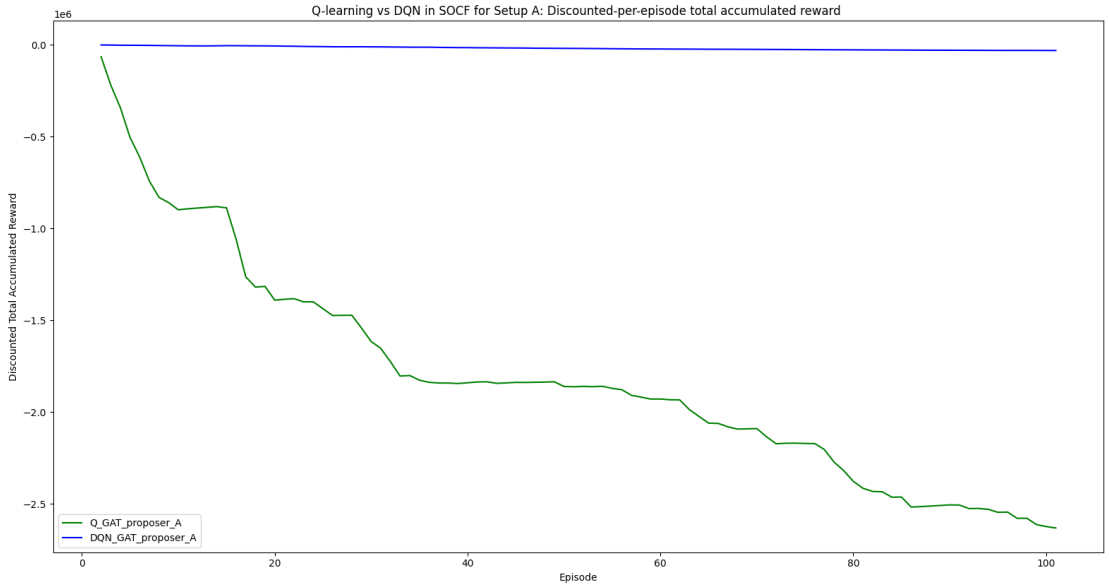

Figure 5.15: Q-learning vs DQN on GAT proposer, Setup A: Discounted-per-episode total accumulated reward

Then, for Setup B:

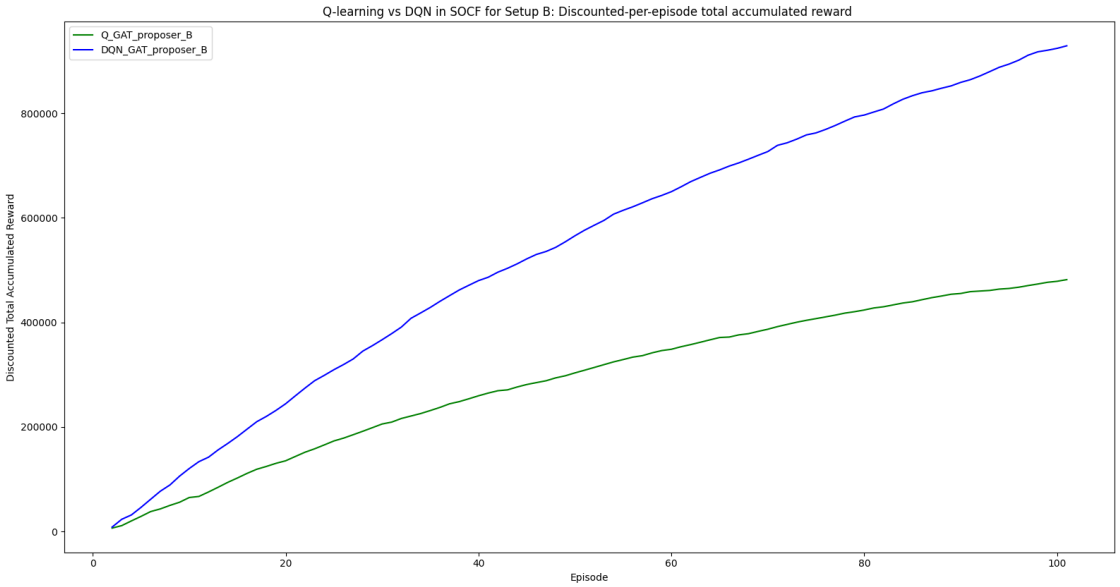

Figure 5.16: Q-learning vs DQN on GAT proposer, Setup B: Discounted-per-episode total accumulated reward

From the above figures, we can clearly see that DQN yields better results and performance against $Q-learning$. Even in the case of Setup A, where both implementations produce negative rewards, DQN is fairing significantly better.

To provide, an even clearer comparison between the two, we show their total results in time and reward in the below table:

Table 5.6: Summary Results for Q vs DQN best combinations

Q vs DQN	**Total Duration**	**Total Reward**
Q Setup A	1,386.385289 sec	-7,097,131.254
DQN Setup A	28.22000432 sec	-119,893.9755
Q Setup B	2,480.478861 sec	57,299.96232
DQN Setup B	1,523.357206 sec	71,742.2617
Q GAT_proposer A	870.2227089 sec	-3,818,542.267
DQN GAT_proposer A	53.39349976 sec	-49,068.47362
Q GAT_proposer B	1,604.18417 sec	755,879.9084
DQN GAT_proposer B	908.72119 sec	1,486,165.278

5.5 Discussion

Our experiments with the mentioned combinations of update methods and algorithms, provide us with some interesting findings to discuss.

We notice, that in both the use of $Q-learning$ and DQN, the importance of when the moment to commence the update remains a crucial part, for the performace. Since, in both protocols, updating the cooperation values whenever a proposer changes yields better rewards than its counterpart where the update is at the end of an episode. This highlights the importance of the temporal placement of the update of cooperation values among the agent types.

Another important observation we can make from our experimental results, is the superiority of the GAT protocol to the *Simple*. This is since, regardless of the algorithm used, we notice the GAT combinations having much better times to using *Simple* method. Even in the case of Setup B, where the environment is of a bigger scale, we can see both $Q-learning$ and DQN being almost twice as fast to their *Simple* counterparts. This demonstrates the power of GATs, in our setting.

Additionally, the GAT update method proves to be remarkably more beneficial,

given the results in the Setup B, in both algorithms, as we clearly see in the Tables 5.2, 5.4. Thus, GAT is both faster and better at handling a Sequential Overlapping Coalition Formation multi-agent environment.

Now, it is important to shine light upon the question, which algorithm is better, $Q-learning$ or DQN. From all the above experiments, it has clearly been shown in both terms of duration and reward, that the DQN outperforms the $Q-learning$ algorithm.

Even in the case of Setup A, where both algorithms have poor results, DQN shows a notably smaller loss than $Q-learning$. In Setup B, their difference is even more apparent, as we can see in their results in Table (5.6), where we compare the best combination for each algorithm for each setup. In the case of Setup B, we notice DQN being twice as profitable at around half the time of $Q-learning$.

It seems that DQN is better suited for our Sequential Overlapping Coalition Formation setting, making the approach of employing neural networks in such cases, an appealing one. The use of DQN's neural network alongside the experience replay mechanism is the cause for the difference in performace, both in time but more importantly on the results in terms of reward. As such, it helps the agents be more well informed about the outcomes of their actions and their choices. This leads to agents making more profitable proposals, as well as, the responders being more precise on their estimation of the proposal's significance, improving their decision-making skills.

DQN is shown to be able to handle a very large action space, in Setup B, and managing to outperform the $Q-learning$ in all combinations and both settings.

In conclusion, we can say that the "clear winner" is the combination of DQN with the GAT update method when a proposer changes. It outperforms in all aspects any of the other combinations we have tested and presented, being evident in both the Figure (5.16) and Table (5.6).

Additionally, GAT shows great potential, being able to handle the uncertainty of the cooperation values DoC, quite well, while also handling a graph representation of a large multi-agent environment like our second setup, with ease. This showcases

the great potential GNNs hold for further use and experimentation in Sequential and Overlapping Coalition Formation settings.

Chapter 6

Conclusions and Future Work

In summary, this thesis has undertaken a comprehensive exploration into the realm of Deep Reinforcement Learning (DRL) and Overlapping Coalition Formation. Through our systematic utilization of $GNNs$ and DRL for our Sequential Overlapping Coalition Formation approach, we sought to battle the uncertainty of such a setting, regarding the cooperation values between the agent types.

The findings of this study have unearthed valuable insights in the fields of DRL, Overlapping Coalition Formation and the use of $GNNs$ in such environments.

One of the primary contributions of this work lies in its examination of the combination of DRL algorithms in a multi-agent environment where sequential overlapping coalition formation takes place. The results, as presented in Chapter 5, highlighted that the integration of DRL is more beneficial than the traditional RL algorithms, significantly advancing our understanding of both DRL and Overlapping Coalition Formation.

Additionally, in our approach we explored the use of GNN to address the specific type of uncertainty of our environment. As it is clear from its combination with RL and DRL algorithms during our research, we notice from the results it yielded in both cases of $Q-learning$ and DQN, as shown in Tables 5.2, 5.4, that it displays better perfomance in terms of speed and payoffs in comparison to a more standard approach like the *simple* update protocol for dealing with our uncertainty. Which agrees with the complexity analysis of both update methods as we discussed previously in Section 4.2.2.

As we have elaborated in Chapter 4 and 5 about our approach on the matter and the results we had, we make the interesting find that DRL and GNN work quite well together in a setting of sequential overlapping coalition formation, which has, to our knowledge, not been tested before this work.

It is essential to acknowledge the limitations of this study, such as the testing of just one DRL algorithm and just one GNN method in our setting. As well as the fact that the way we incorporate the GNN, we provide the agents with all the information about the DoC among agent types, though each agent only exploits information relating to the DoCs concerning its own agent type only. Another limitation is the fact that our experiments and testing was on the two settings of the environment's parameters mentioned earlier. We encourage further experimentation on the values of those parameters, and testing of the performance of the algorithms and update methods on an even larger scale.

These constraints, while inherent in the research process of a thesis, provide avenues for future exploration. Researchers in the field may benefit from addressing these limitations to refine and expand upon the current work.

Looking forward, there are several promising directions for future research. Testing other DRL algorithms, (for instance, state of the art "deep exploration" RL algorithms) alongside different GNN variants for the update of DoC values, present exciting opportunities to delve deeper into both fields in the scope of sequential overlapping coalition formation. One could also, test GNNs with another non-DRL decision making approach for the deliberations of the agents (eg. an "intelligent search" one, such as Monte Carlo Tree Search) These recommendations aim to build upon the foundation laid by this thesis and contribute to the ongoing discource within the field.

Another interesting direction future research could take, is to make an evaluation of the performance of specific agent types. For instance "strong" types that are able to form more rewarding cooperation bonds with other agents; and also examine learning in setting where the types of agents are unknown, and the learner has to infer them via observing the actions and perfomance of others as they form coalitions.

Further research could focus on the use of GNN as a means of updating the information of agents. Specifically, an agent could be updated about the DoC values for the types of agents it has encountered or for the k most "high valued" DoC, where $k \in \mathbb{N}$. In this sense, the GNN could potenitally work as a trusted third-party informing the agents, possibly leading to stable rational outcomes in the spirit of correlated equilibria [4].

In conclusion, this research has demonstrated the importance of DRL and its potential impact on Overlapping Coalition Formation alongside the employment of $GNNs$. The synthesis of our findings, limitations, and future recommendations serve as a comprehensive reflection on the journey of this thesis. As we move forward, it is our hope that this work may act as a catalyst for further investigations, fostering a deeper understanding of DRL and Overlapping Coalition Formation.

Bibliography

[1] Paul Jaccard. "THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1". In: *New Phytologist* 11.2 (1912), pp. 37–50. DOI: https://doi.org/10.1111/j.1469-8137.1912.tb05611.x. eprint: https://nph.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-8137.1912.tb05611.x. URL: https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x.

[2] RICHARD BELLMAN. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684. ISSN: 00959057, 19435274. URL: http://www.jstor.org/stable/24900506.

[3] Peter J. Huber. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. DOI: 10.1214/aoms/1177703732. URL: https://doi.org/10.1214/aoms/1177703732.

[4] Robert J. Aumann. "Correlated Equilibrium as an Expression of Bayesian Rationality". In: *Econometrica* 55.1 (1987), pp. 1–18. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1911154.

[5] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: *Machine Learning* 8.3 (May 1992), pp. 279–292. ISSN: 1573-0565. DOI: 10.1007/BF00992698. URL: https://doi.org/10.1007/BF00992698.

[6] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8 (1992), pp. 229–256.

[7] Francis Bloch. "Sequential Formation of Coalitions in Games with Externalities and Fixed Payoff Division". In: *Games and Economic Behavior* 14.1

(1996), pp. 90–123. ISSN: 0899-8256. DOI: https://doi.org/10.1006/game.1996.0043. URL: https://www.sciencedirect.com/science/article/pii/S0899825696900433.

[8] Onn Shehory and Sarit Kraus. "Methods for task allocation via agent coalition formation". In: *Artificial Intelligence* 101.1 (1998), pp. 165–200. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00045-9. URL: https://www.sciencedirect.com/science/article/pii/S0004370298000459.

[9] Bastian Blankenburg, Matthias Klusch, and Onn Shehory. "Fuzzy kernel-stable coalitions between rational agents". In: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (Jan. 2003), p. 9. DOI: 10.1145/860576.860578.

[10] 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, 2004. AAMAS 2004.* (2004), pp. 1090–1097.

[11] Lovekesh Vig and Julie A. Adams. "Coalition Formation: From Software Agents to Robots". In: *Journal of Intelligent and Robotic Systems* 50 (2007), pp. 85–118.

[12] Georgios Chalkiadakis and Craig Boutilier. "Sequential Decision Making in Repeated Coalition Formation under Uncertainty". In: AAMAS '08 (2008), pp. 347–354.

[13] Georgios Chalkiadakis, Edith Elkind, Evangelos Markakis, and Nicholas R. Jennings. "Overlapping Coalition Formation". In: *Internet and Network Economics* (2008). Ed. by Christos Papadimitriou and Shuzhong Zhang, pp. 307–321.

[14] W.T. Luke, George Chalkiadakis, and Teacy et al. "Sequential Decision Making with Untrustworthy Service Providers". In: AAMAS '08 (2008), pp. 755–762.

[15] Martin J. Osborne. *Introduction to Game Theory: International Edition.* OUP Catalogue 9780195322484. Oxford University Press, 2009, pp. 282–294.

[16] G. Chalkiadakis, E. Elkind, E. Markakis, M. Polukarov, and N. R. Jennings. "Cooperative Games with Overlapping Coalitions". In: *Journal of Artificial Intelligence Research* 39 (Sept. 2010), pp. 179–216. DOI: 10.1613/jair.3075. URL: https://doi.org/10.1613%2Fjair.3075.

[17] Nicole Bäuerle and Ulrich Rieder. *Markov Decision Processes with Applications to Finance*. Jan. 2011. DOI: 10.1007/978-3-642-18324-9.

[18] Georgios Chalkiadakis, Valentin Robu, Ramachandra Kota, Alex Rogers, and Nick Jennings. "Cooperatives of Distributed Energy Resources for Efficient Virtual Power Plants". In: *The Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011) (02/05/11 - 06/05/11)* (2011). Event Dates: 2-6 May 2011, pp. 787–794. URL: https://eprints.soton.ac.uk/271950/.

[19] Georgios Chalkiadakis and Craig Boutilier. "Sequentially optimal repeated coalition formation under uncertainty". In: *Autonomous Agents and Multi-Agent Systems* 24 (May 2012). DOI: 10.1007/s10458-010-9157-y.

[20] Georgios Chalkiadakis, Evangelos Markakis, and Nicholas R. Jennings. "Coalitional Stability in Structured Environments". In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '12 (2012), pp. 779–786.

[21] Martijn Otterlo and Marco Wiering. "Reinforcement Learning and Markov Decision Processes". In: *Reinforcement Learning: State of the Art* (Jan. 2012), pp. 3–42. DOI: 10.1007/978-3-642-27645-3_1.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with Deep Reinforcement Learning". In: *NIPS Deep Learning Workshop 2013* (2013). arXiv: 1312.5602 [cs.LG].

[23] Yair Zick Georgios Chalkiadakis, Edith Elkind, and Evangelos Markakis. "Cooperative Games with Overlapping Coalitions: Charting the Tractability Frontier". In: *Artificial Intelligence (AIJ)* (2014). arXiv: 1407.0420 [cs.GT].

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning Workshop* (2015). arXiv: 1503.02531 [stat.ML].

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836. URL: http://dx.doi.org/10.1038/nature14236.

[26] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. "Trust Region Policy Optimization". In: (2015). Ed. by Francis Bach and David Blei. arXiv: 1502.05477 [cs.LG].

[27] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[28] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-learning". In: AAAI'16 (2016), pp. 2094–2100.

[29] Filippo Bistaffa, Alessandro Farinelli, Georgios Chalkiadakis, and Sarvapali D. Ramchurn. "A cooperative game-theoretic approach to the social ridesharing problem". In: *Artificial Intelligence* 246 (2017), pp. 86–117. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2017.02.004. URL: https://www.sciencedirect.com/science/article/pii/S0004370217300243.

[30] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. "On Calibration of Modern Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Proceedings of Machine Learning Research 70 (Aug. 2017). Ed. by Doina Precup and Yee Whye Teh.

[31] Dhouha ben noureddine, Atef Gharbi, and Samir Ahmed. "Multi-agent Deep Reinforcement Learning for Task Allocation in Dynamic Environment". In:

International Conference on Software and Data Technologies (Jan. 2017), pp. 17–26. DOI: 10.5220/0006393400170026.

[32] Aasheesh Singh. "An Object-oriented approach to Robotic planning using Taxi domain". In: (2017). arXiv: 1701.04350 [cs.RO].

[33] Vincent Franois-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and JoellePineau. "An Introduction to Deep Reinforcement Learning". In: *Foundations and Trends® in Machine Learning* 11.3-4 (2018), pp. 219–354. DOI: 10.1561/2200000071. URL: https://doi.org/10.48550/arXiv.1811.12560.

[34] Kyowoon Lee, Sol-A Kim, Jaesik Choi, and Seong-Whan Lee. "Deep Reinforcement Learning in Continuous Action Spaces: a Case Study in the Game of Simulated Curling". In: Proceedings of Machine Learning Research 80 (2018). Ed. by Jennifer Dy and Andreas Krause, pp. 2937–2946. URL: https://proceedings.mlr.press/v80/lee18b.html.

[35] Michalis Mamakos and Georgios Chalkiadakis. "Overlapping Coalition Formation via Probabilistic Topic Modeling". In: AAMAS '18 (2018), pp. 2010–2012.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018, pp. 323–337.

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, and Adriana Romero et al. "Graph Attention Networks". In: *International Conference on Learning Representations* (2018). arXiv: 1710.10903 [stat.ML].

[38] C. P. Andriotis and K. G. Papakonstantinou. "Managing engineering systems with large state and action spaces through deep reinforcement learning". In: *Reliability Engineering and System Safety* 191 (2019), p. 106483. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2019.04.036. URL: https://www.sciencedirect.com/science/article/pii/S0951832018313309.

[39] "Continual learning of context-dependent processing in neural networks". In: *Nature Machine Intelligence* (2019). DOI: 10.1038/s42256-019-0080-x. URL: https://doi.org/10.1038%2Fs42256-019-0080-x.

[40] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. "A survey and critique of multiagent deep reinforcement learning". In: *Autonomous Agents and Multi-Agent Systems* 33.6 (Oct. 2019), pp. 750–797. DOI: 10.1007/s10458-019-09421-1. URL: https://doi.org/10.1007%2Fs10458-019-09421-1.

[41] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. "Q-Learning Algorithms: A Comprehensive Classification and Applications". In: *IEEE Access* 7 (2019), pp. 133653–133667. DOI: 10.1109/ACCESS.2019.2941229.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[43] Mohit Sewak and Mohit Sewak. "Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence". In: *Deep Reinforcement Learning: Frontiers of Artificial Intelligence* (2019), pp. 95–108.

[44] Chloe Ching-Yun Hsu, Celestine Mendler-Dünner, and Moritz Hardt. "Revisiting design choices in proximal policy optimization". In: *arXiv preprint arXiv:2009.10897* (2020).

[45] Alec Koppel, Garrett Warnell, Ethan Stump, Peter Stone, and Alejandro Ribeiro. "Policy evaluation in continuous MDPs with efficient kernelized gradient temporal difference". In: *IEEE Transactions on Automatic Control* 66.4 (2020), pp. 1856–1863.

[46] Songming Liu and Shaojun Zeng. "Quantitative Analysis Method of Immunochromatographic Strip Based on Reinforcement Learning". In: *Journal of Physics:*

Conference Series 1449 (Jan. 2020), p. 012058. DOI: 10.1088/1742-6596/1449/1/012058.

[47] Teng Liu, Bing Huang, Xingyu Mu, Fuqing Zhao, Xiaolin Tang, and Dongpu Cao. "A Comparative Analysis of Deep Reinforcement Learning-enabled Freeway Decision-making for Automated Vehicles". In: *arXiv preprint arXiv:2008.01302* (2020).

[48] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. "Multi-agent game abstraction via graph attention neural network". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (2020), pp. 7211–7218.

[49] Pei-Hsin Wang, Sheng-Iou Hsieh, Shih-Chieh Chang, Yu-Ting Chen, Jia-Yu Pan, Wei Wei, and Da-Chang Juan. "Contextual temperature for language modeling". In: *arXiv preprint arXiv:2012.13575* (2020).

[50] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[51] C.P. Andriotis and K.G. Papakonstantinou. "Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints". In: *Reliability Engineering & System Safety* 212 (2021), p. 107551. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.107551. URL: https://www.sciencedirect.com/science/article/pii/S095183202100106X.

[52] Shushman Choudhury, Jayesh K Gupta, Peter Morales, and Mykel J Kochenderfer. "Scalable anytime planning for multi-agent mdps". In: *arXiv preprint arXiv:2101.04788* (2021).

[53] Yannan Li, Yong Yu, Willy Susilo, Zhiyong Hong, and Mohsen Guizani. "Security and Privacy for Edge Intelligence in 5G and Beyond Networks: Challenges

and Solutions". In: *IEEE Wireless Communications* 28 (Apr. 2021), pp. 63–69. DOI: 10.1109/MWC.001.2000318.

[54] Hannan Amoozad Mahdiraji, Elham Razghandi, and Adel Hatami-Marbini. "Overlapping coalition formation in game theory: A state-of-the-art review". In: *Expert Systems with Applications* 174 (2021), p. 114752. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.114752. URL: https://www.sciencedirect.com/science/article/pii/S0957417421001937.

[55] Elise van der Pol, Herke van Hoof, Frans A Oliehoek, and Max Welling. "Multi-agent mdp homomorphic networks". In: *arXiv preprint arXiv:2110.04495* (2021).

[56] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[57] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, and Mathieu Reymond et al. "A practical guide to multi-objective reinforcement learning and planning". In: *Autonomous Agents and Multi-Agent Systems* 36.1 (2022). DOI: 10.1007/s10458-022-09552-y. URL: https://doi.org/10.1007%2Fs10458-022-09552-y.

[58] Sahand Rezaei-Shoshtari, Rosie Zhao, Prakash Panangaden, David Meger, and Doina Precup. "Continuous MDP Homomorphisms and Homomorphic Policy Gradient". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 20189–20204.

[59] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. "Gstarx: Explaining graph neural networks with structure-aware cooperative games". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19810–19823.

[60] Mikko Lauri, David Hsu, and Joni Pajarinen. "Partially Observable Markov Decision Processes in Robotics: A Survey". In: *IEEE Transactions on Robotics* 39.1 (Feb. 2023), pp. 21–40. DOI: 10.1109/tro.2022.3200138. URL: https://doi.org/10.1109%2Ftro.2022.3200138.

[61] Fangzhu Ming, Feng Gao, Kun Liu, and Chengmei Zhao. "Cooperative modular reinforcement learning for large discrete action space problem". In: *Neural Networks* 161 (2023), pp. 281–296. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2023.01.046. URL: https://www.sciencedirect.com/science/article/pii/S0893608023000588.

[62] James Orr and Ayan Dutta. "Multi-agent deep reinforcement learning for multi-robot applications: a survey". In: *Sensors* 23.7 (2023), p. 3625.

[63] 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". In: *IEEE Transactions on Mobile Computing* 22.08 (Aug. 2023), pp. 4439–4455. ISSN: 1558-0660. DOI: 10.1109/TMC.2022.3165965.