UTS UNIVERSITY OF TECHNOLOGY SYDNEY

A Decentralized Multi-Agent Online Planning Algorithm and Application on Smart Grid

by Wenhao Zhuo

Thesis submitted in fulfilment of the requirements for the degree of

Master of Analytics (Research)

under the supervision of A/Prof. Jianlong Zhou, A/Prof. Zhidong Li, and Prof. Fang Chen.

University of Technology Sydney
Faculty of Engineering and Information Technology

August 2025

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Wenhao Zhuo, declare that this thesis is submitted in fulfilment of the requirements for the award of *Master of Analytics*, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 19/08/2025

Abstract

In response to the worldwide long-term emissions reduction plan, more distributed renewable resources, energy storage systems, and electric vehicles are expected to be included in the power grid for the next few decades. However, achieving high penetration of renewable energy can still be challenging because of the inherent variability and intermittency of renewable energy resources. Managing distributed energy storage units and electric vehicles also produces many high-dimensional and multi-type data, which can be challenging with conventional modeling and control techniques. Therefore, an intelligent reconfiguration of the existing power grid is needed. One promising solution is to decompose the entire power grid into many decentralized and distributed microgrids that have independent power supplies with renewable energy resources. The management of these microgrids could be formulated as multi-agent reinforcement learning problems, with objectives such as maximizing profit gained by energy or minimizing the electricity cost. Agents include controllers for distributed renewable resources, energy storage systems, electric vehicle charging stations, and, in general, consumers who can influence the power grid.

Given this background, our research aims to address some of the decision-making problems in smart grid systems with multi-agent reinforcement learning, focusing on decentralized and partially observable environments that reflect uncertainties in renewable forecasting, load demand, and rewards. Our main contribution is a novel decentralized multi-agent planning algorithm that is built upon previous research on point-based POMDP solvers, rollout, self-learning, upper confidence bounds search, and common information approach. In particular, it can be applied to decentralized stochastic control problems with a given model for training the policy offline and designing the corresponding online/real-time planning policy. It is suitable and easy to implement under the "off-line training, on-line play" framework, which has demonstrated excellence in many state-of-the-art artificial intelligence systems.

ACKNOWLEDGMENTS

I wish to first thank my principle supervisor Prof. Jianlong Zhou for his endless support in completing this thesis, especially when I was hospitalized earlier this year. It took a lot of efforts for me to recover from the illness, and during that period, he insisted me to prioritize my health, and return to research when I am fully recovered. The ideas and suggestions provided by my co-supervisor Prof. Zhidong Li also helped me a lot when writing this thesis, he insisted me on contributing novel ideas in RL rather than applications of RL, that pushed me forward.

I want to express my heartfelt gratitude to my parents for their unwavering belief in me and for encouraging me to pursue a degree in a different field, when I was severely sick and unconscious in the hospital, they were always on my side to guard me. Your support has meant the world to me. And I also want to thank Dr. Alice Grey and Dr. Leon Edwards for saving my life, I cannot imagine to recover so well without their help and support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

Cooperative multi-agent reinforcement learning (MARL) algorithms can train multiple decision-making agents to cooperate and collaborate in a stochastic environment, with the purpose of maximizing a joint cumulative reward. Many practical applications can be formulated as MARL problems, and it is a dynamic and interdisciplinary field, continually advancing with the development of new algorithms and applications.

Due to the worldwide long-term emissions reduction plan, the existing power grid is expected to experience a transformation in the coming decades, with more and more distributed renewable resources, energy storage systems, and electric vehicles being integrated. As more households choose to install small-scale distributed renewable resources and energy storage units (such as battery storage and fast-response thermostatically controlled loads), consumers are transitioning into prosumers that are capable of producing and storing electricity. In countries with deregulated electricity markets, these prosumers can participate in the market via peer-to-peer energy trading and demand response programs, which could reduce their electricity cost with proper control of the renewable power they produce and store. Therefore, there is a growing need to incorporate intelligent and bidirectional infrastructure to accommodate this utility service provider and their customers, as well as to develop new control and management strategies.

From the perspective of MARL, the power grid acts as the environment in which these customers interact and collaborate. These customers can be considered as intelligent agents that make decisions to maximize certain rewards, such as profit gained by energy trading and the electricity costs saved with efficient use of green energy. Also, the increasing integration of distributed renewable energy introduces greater uncertainties in power grid operations due to the stochastic nature and intermittency of renewable energy sources. The management of distributed energy resources and charging stations of electric vehicles can be challenging as they rely heavily on forecasting data to operate and can produce a large amount of high-dimensional and multi-type data. Therefore, a learning-based technique fits naturally well for a smart grid compared to the central and unidirectional control paradigm in the existing power grid.

Nevertheless, a major issue of reinforcement learning is scalability, and it is unrealistic to formulate the entire power grid as a MARL problem that includes millions of agents and an astronomically large number of states/observations. Fortunately, this problem can be partially mitigated with microgrids. These small-scale electrical distribution networks can operate in both grid-connected and islanded modes, which could be one of the promising solutions to increase the penetration of renewable power. Effective management of energy storage systems, using renewable energy and load forecasts, could help stabilize renewable energy, making it available for use on demand within these microgrids. It is also believed that the paradigm shift in microgrids could accelerate the massive integration of distributed renewable energy sources without the need for a complex centralized coordination network, thus supporting the development and realization of future smart grid systems.

As more microgrids are developed to enhance renewable energy integration, the interconnection of multiple microgrids is emerging as an advanced operational paradigm, enhancing the resiliency and reliability of power system networks. A networked microgrid system comprises physically interconnected and functionally interoperable microgrids. With proper control and communication technologies, microgrids within a network can be coordinated at a higher level to enable mutual power sharing with neighboring microgrids, provide advanced ancillary services, deliver operational benefits for stakeholders, and maintain a more reliable and economical power supply.

As a result, a distributed and decentralized MARL algorithm would be necessary to

control the next-generation power grid system. A network of many self-governed and intelligent microgrids with prosumers, designed to collaborate effectively, will be capable of accommodating high levels of renewable energy integration by leveraging advancements in forecasting, distributed energy resources, battery energy storage systems, electric vehicles, peer-to-peer energy trading, and demand response. MARL algorithms will be the key to optimizing cooperation in a stochastic environment, leading to a greener future.

1.2 Research Problems

1.2.1 Overview

This research primarily centers on on-line planning algorithms within decentralized multi-agent cooperative settings. The main contribution of this thesis is the development of a novel planning algorithm capable of addressing decentralized multi-agent planning problems in both off-line and real-time scenarios. Furthermore, we carried out a case study to apply our algorithm to a microgrid management problem and introduced an innovative battery aging cost model.

In many domains, agents possess limited or noisy sensing capabilities, preventing them from directly accessing the state signals of the environment. As a result, many problems are modeled as partially observable Markov decision processes (POMDPs). With instantaneous and perfect communication, agents can maintain a joint belief over the states, which is a probability distribution reflecting the true state values based on the history of previous observations and/or actions. This joint belief serves as a sufficient statistic for decision making and can be updated using Bayesian methods, such as Kalman filtering or particle filtering [7]. In this scenario, the controller or policy of each agent maps the belief state to its actions. In addition, a central cloud system is necessary to gather observation data from all agents, calculate the joint belief, and transmit it back to the agents.

In real-world scenarios, free and instantaneous communication is often unavailable, making the decentralized partially observable Markov decision process (Dec-POMDP) model more suitable for most multi-agent environments. In this model, agents rely on local observations to take actions that maximize a joint reward, with each agent's policy

mapping its local observation history to actions. Since direct sharing of previous observations and actions is often restricted during operation, it becomes impractical to maintain a shared joint belief state across all agents. This decentralized and partially observable setting significantly increases the complexity of solving Dec-POMDP problems optimally. As noted in [6], solving an optimal solution for a finite-horizon Dec-POMDP involving more than two agents is NEXP-complete, with a doubly exponentially growing number of joint policies.

The decentralized multi-agent setting poses a significant challenge to achieving one critical control paradigm: enabling agents to perform on-line planning during real-time operation. Typically, algorithms designed to solve Dec-POMDP problems adopt the centralized training for decentralized execution (CTDE) approach. In the centralized training phase, agents have access to extensive information about the environment, including the dynamics, rewards, and conditions of other agents. These algorithms use this information to train a policy tree, finite state controller, or recurrent neural network, which maps sequences of agents' observations and actions to appropriate actions based on their histories. However, once deployed in the actual environment, agents rely solely on the policies trained offline, making real-time re-planning generally infeasible in a fully decentralized setting.

1.2.2 Research Aim

As multiple agents performing on-line planning in a decentralized environment could be infeasible, they can only base their decisions on the off-line trained policies during execution. In general, the optimal policies for a large-scale decentralized multi-agent problem are intractable to solve, which is due to the fact that Dec-POMDP models have doubly exponentially growing joint policy space, as well as the inherent errors within approximation frameworks such as neural networks and finite state controllers. Also, the simulator used in offline training is unlikely to be a perfect representation of the practical environment/real-world. Therefore, most controllers/policies solved off-line for complex multi-agent problems could be suboptimal, and further improvements would require an extensively more collection of data and fine-tuning on parameters. On the other hand, off-line trained policies could be enhanced by adopting them "smartly" during execution, which is discovered by the research monograph in [9] and is referred to as "off-line training, online play". Nevertheless, in a decentralized setting, this type

of run-time modification of plan-time policies cannot be achieved, as we discussed in the previous section. Drawing inspiration from the results in [9], we aim to fill this gap by developing an "on-line play policy" suitable for multiple agents operating in a decentralized environment with restricted communication.

The conceptual framework of "off-line training, on-line play" for control and reinforcement learning was first introduced in [9], drawing significant inspiration from the successes of the AlphaZero program (designed to play chess) and the TD-Gammon program (designed to play backgammon), as noted by the author. In this framework, the "off-line training" phase involves an algorithm learning to evaluate game positions and make moves that lead to advantageous outcomes against a default player. The "on-line play" phase uses the trained offline player to generate high-quality moves in real-time during matches against opponents. The author observed that the on-line players in AlphaZero and TD-Gammon outperformed their extensively trained off-line counterparts, and emphasized that the performance of policies trained offline can be significantly enhanced through on-line play.

Specifically, a purely off-line trained policy for a complex Dec-POMDP model with many agents could degrade in a complex and dynamic environment, which could be caused by inaccurate approximations that are inherent in feature-based predictors such as neural networks. According to the author, the "on-line play policy" functions as a lookahead optimization combined with rollout simulation, using a simple or extensively trained base policy, supplemented by an (optional) cost approximation at the end of rollout simulation. The author theorizes that the performance improvement occurs because on-line play effectively performs a step in Newton's method for solving Bellman's equation. This approach could leverage the superlinear convergence property of Newton's method, potentially enhancing the suboptimal off-line trained policy and driving it closer to optimality.

This performance enhancement by an "on-line play policy" is a broadly applicable paradigm for many reinforcement learning, optimization, and control problems. It is also well suited for situations where the environments are open and dynamic, which can have varying rules, team members, and dynamical models, such as in the context of ad hoc teamwork [44] and adaptive control problems. Nevertheless, as stated earlier, this on-line lookahead optimization cannot be performed in a fully decentralized, partially

observable multi-agent environment.

1.2.3 Main Contribution

Based on our review of recent research on applications of MARL in power grids, we noticed that most of them only focus on developing purely off-line algorithms, and little has been done on the on-line planning part. In addition, many of the algorithms are centralized, assuming instantaneous and perfect communication. This is not surprising given the complexity of a power grid management problem, which would involve much larger spaces than standard benchmark problems. Considering a decentralized setting in their problems would worsen the complexity and could become intractable to solve in both simulation and real-world. We also discovered that an exact implementation of the "off-line training, on-line play" paradigm [9] in a fully decentralized setting could be infeasible/intractable, which is due to the infinite hierarchy of belief reasoning. For instance, one agent must construct a belief on the other agents' private information, as well as the belief on other agents' belief on its own private information, and so on. Therefore, even for small toy problems with few agents, the joint policy/belief space needed in "on-line play" (i.e., for rollout simulation and lookahead optimization) will grow doubly exponentially and would be infeasible to implement. As a result, we enable the "on-line play" policies in decentralized environments approximately, by assuming a decentralized/implicit communication that is referred to as the common information assumption. And our main contribution is an "on-line play policy" that can be applied in a decentralized multi-agent setting under a common information assumption.

Originally introduced in [35], the common information approach is a framework to tackle multi-agent decision-making problems in which agents operate independently, with limited observability and without centralized coordination. In contrast to a fully decentralized setting, where each agent has access only to its private information, this approach leverages shared common information that is accessible to all agents, such as a delayed sharing of observations or historical information, to decompose the overall decision-making problem and simplify the control process.

The approach reduces the complexity of decentralized control with the help of a shared common information, which enables each agent to concentrate on its private information while coordinating indirectly via a shared observation. This framework

allows for a structured reformulation of the problem, converting a Dec-POMDP into a coordinator's partially observable Markov decision process (CPOMDP) for each agent, where decisions are based on a combination of common and private information. Specifically, it is assumed that a virtual coordinator observes the common information and provides "prescriptions" to each agent. These "prescriptions" are mappings that translate an agent's private information into its actions, without granting the coordinator access to the agents' private information. As a result, the control strategy remains decentralized.

Nevertheless, this type of common information may not always exist and needs to be created via a partial/delayed/noisy sharing of private information. It would somewhat violate the standard conditions of a Dec-POMDP model and incur additional costs, such as additional communication protocols must be designed for sharing information. We believe that the additional costs could be compensated by the benefits brought by our "on-line play" policy.

1.3 Structure of Thesis

In the next chapter, we provide a comprehensive literature review covering topics on reinforcement learning, online planning algorithms, common information approaches, decentralized partially observable MDP, and their applications in smart grids, along with an analysis of research gaps. Chapter 3 provides a detailed explanation to our decentralized multi-agent planning algorithm, followed by a case study on a microgrid management problem in Chapter 4. Chapter 5 includes a discussion of potential ideas that could improve our algorithm and scale to complex and dynamic systems. The thesis concludes with an exploration of future research directions in the final section.

CHAPTER 2

LITERATURE REVIEW

2.1 A Brief Overview on Reinforcement Learning

Reinforcement Learning (RL) is primarily grounded in two essential components: the agent and the environment. The entire field of RL research revolves around the concept of computational learning through interactions between these two entities, as seen in Figure 2.1.

Environment can be considered as a system that includes the basic components of an optimization problem, such as a state reflecting the current status of interests, an interface for agent interaction, and the capability to update its state based on interactions or internal processes. **Agent** is a decision-making entity tasked with solving an optimization problem or achieving specific objectives within the environment. In addition to these two fundamental elements, a reinforcement learning (RL) system comprises four main sub-elements: a policy for action selection, a reward signal, a value function and, optionally, a model of the environment [58].

Policy can be viewed as a mapping from the states of the environment to the actions of an agent, which also represents the agent's behavior in the learning process. It can take the form of a simple function, a lookup table, or a complicated search process. Also, in scenarios where the actual state signal of the environment is not always/partially accessible to agents, such as in a POMDP, the policy may instead take other inputs, such

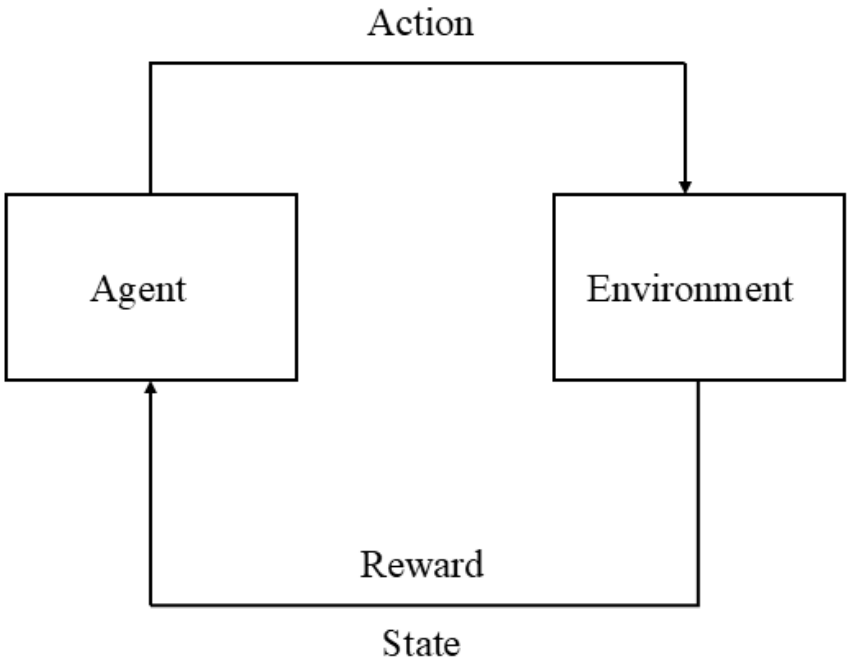

Figure 2.1: Basic Reinforcement Learning

as a sequence of private histories or a probability distribution over states. In addition, the policy could be stochastic, reflecting the probabilities of each action.

Reward, or cost serves as a critical metric for assessing the effectiveness of a learned policy and establishes the objective in an RL problem. Typically, each state transition produces a corresponding reward or cost, and the agent's goal is to maximize the cumulative reward/minimize the additive cost over a planning horizon that can be finite or discounted infinite. The reward signal acts as the main mechanism for defining the policy. If a policy-selected action yields a low reward, the policy can be adjusted to choose a different action in similar situations in the future. In general, reward signals are often stochastic functions of states and actions, defined according to problem-specific parameters. For example, in robotics navigation problems, the reward might be determined by the proximity to the target location, while in repair problems, the cost could reflect the number and severity of damaged sites. It is important to recognize that rewards and costs are essentially opposite signals that differ only in their signs. Designing an appropriate reward function for each time step is a key element in formulating an RL problem.

The reward signal offers immediate feedback on what is beneficial, while the **value function** captures the long-term outcome of a state by accounting for the future trajectory of states achievable under a given policy. Essentially, the value of a state reflects the total expected reward an agent can accumulate in the long run, starting from that state. Although rewards indicate the short-term desirability of a state, the value function assesses its overall desirability by incorporating future states and their associated rewards or costs. For example, a state might consistently yield a low immediate reward (or high immediate cost), yet still have a high value (or low cost function value) because it leads to favorable future states. On the other hand, a state may yield high immediate rewards but have a low long-term value if it frequently leads to less favorable states. In this way, the value function bridges short-term rewards or costs with long-term outcomes, and the quality of an RL method is directly tied to the precision of its value function estimation. According to [58], from a human perspective, rewards/costs can be likened to immediate pleasure/pain, while value functions reflect a more thoughtful and forward-looking assessment of how satisfied or dissatisfied we are with the state of our environment.

A **model** is an important but optional component in RL, often used to plan by predicting state transitions within a simulated environment. Although some problems allow the agent to learn through repeated interactions in the real environment without a model, in most practical applications, a model is often crucial. This component simulates the behavior of the environment or allows predictions about how the environment will respond to specific actions. For instance, given a state and an action, the model may predict the subsequent state and the corresponding reward. Models play a central role in planning, which involves determining a long-term outcome of action by considering potential future scenarios before directly experiencing them. RL methods that utilize models and planning are known as model-based methods, in contrast to model-free methods, which depend on explicit trial-and-error learning and are often regarded as the opposite of planning. Modern RL encompasses a wide range of algorithms, ranging from trial-and-error techniques to adaptive and deliberative planning strategies. In the following section, we will provide a brief review of different RL algorithms.

Multi-Agent Reinforcement Learning (MARL) is an extension of the single-agent framework, where both the state and the value function are influenced by the policies of multiple agents rather than just one. Alternatively, actions can be decomposed into

multiple components, each corresponding to an individual agent. In this case, state transitions and environmental rewards are governed by the joint action space, which expands exponentially as the number of agents increases. In addition, each agent must optimize a value function that is influenced by the decisions of other agents, leading to various scenarios in MARL depending on the tasks and the nature of the interactions [1]. **Cooperative Scenarios**: These are the most straightforward and common settings in MARL, where all agents collaborate to optimize a common reward or objective. In such cases, agents optimize their local policies to collectively achieve the global objective. Alternatively, agents may have individual (private) rewards that are aggregated at the global level, enabling a distributed and decentralized setup with limited communication among agents. A **Competitive Scenario** occurs when conflicting objectives are assigned to multiple agents, which is commonly encountered in computer games and has been extensively studied in the context of two-player games. In a fully competitive game, one agent's reward is the inverse of the other agent's reward, meaning that when one agent gains a reward, the other incurs a penalty. So, the optimal performance is achieved when the total reward equals zero, making this framework well-suited for various types of board games, card games, and video games. Lastly, a **Mixed Scenario** can emerge from a combination of cooperative and competitive agents. It can also occur when each agent pursues its own goal, which may or may not conflict with the goals of others. These scenarios are typically modeled as general-sum games.

2.2 Taxonomy of RL algorithms

In this section, we provide an overview of the classic RL algorithms, following the taxonomy presented in [22] and depicted in Figure 2.2. As illustrated in the figure, a key distinction among RL algorithms is whether the agent has access to a model (model-based) or learns the model (model-free) of the environment. Typically, a model refers to a function or simulator that can predict state transitions and associated rewards.

The main advantage of the model-based approach is that it allows planning by simulating possible future scenarios, evaluating the outcomes of different choices, and explicitly selecting the optimal option for agents. This planning process enables the agent to formulate a more effective policy using the results of the simulated scenarios. A prominent example of this approach is AlphaZero [49], which resulted in notable gains

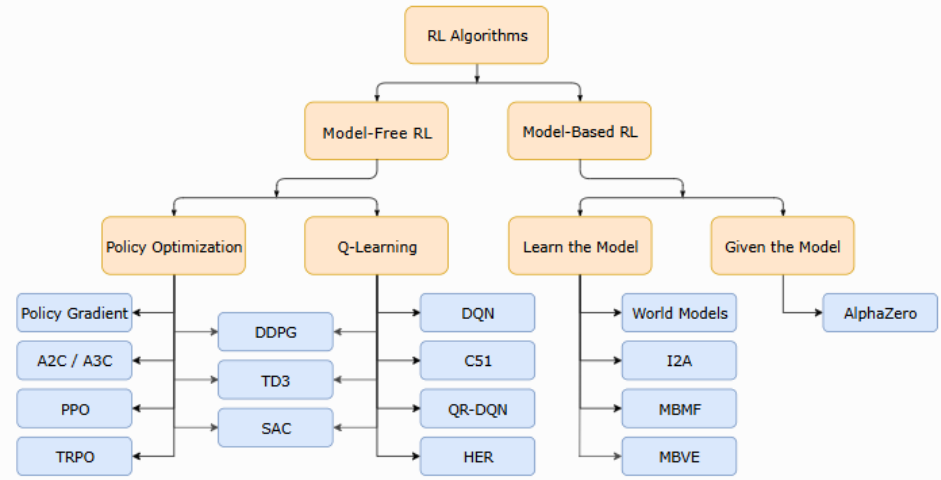

Figure 2.2: A taxonomy of RL algorithms, modified from [22].

in sample efficiency compared to model-free methods. However, a major limitation is that a perfect model reflecting the testing environment is often unavailable. In such cases, the agent must first collect data from its experiences to learn the model, namely finding parameters to fit a neural network to predict a model, which could be difficult in some cases. The most critical issue is that inaccuracies or biases in the learned model can lead to overfitting, where the agent can perform well within the model but poorly when deployed in the practical environment. Furthermore, learning an accurate model is complex and even significant investments in time and computational resources may not yield reliable results.

Algorithms that do not rely on models are referred to as model-free methods, whereas those that utilize models, whether provided or learned, are known as model-based methods. Model-based methods often achieve higher sample efficiency, whereas model-free methods are generally easier to implement and fine-tune. According to [22], model-free reinforcement learning primarily uses two approaches in training: Q-Learning and Policy Optimization.

2.2.1 Q-Learning

In Q-Learning, the main objective is to learn an Q-function $Q_\theta(s,a)$ that serves as an approximator. The optimal value function in state s is usually denoted as $V^*(s)$, and Q-learning aims to approximate the optimal $Q^*(s,a)$ (i.e., the optimal value function by

taking action a at s), by training a feature-based predictor such as a neural network with parameters θ. The optimization typically relies on the Bellman equation and is mostly performed off-policy, which means data from any point during training can be used, regardless of the policy used to collect it. The policy is determined by selecting the best action that optimizes the Q-value:

$$a(s) = \arg\max_a Q_\theta(s,a). \tag{2.1}$$

Deep Q network (DQN) [31] is a foundational algorithm that popularized deep RL. It is well known for its remarkable success in playing Atari games and achieving super human-level performance. Two key methods are incorporated into DQN for learning: experience replay and target network. An experience is defined as a tuple $(s_k, a_k, r_k, s_{k+1},)$. The algorithm selects actions using an $\epsilon-$greedy action based on Q-values at each time step, and the experiences are stored in a memory buffer capable of holding millions of state transition data. This stored data is then used to train a neural network using stochastic gradient descent to minimize the following loss function:

$$L(\theta) = \left[r_k + \gamma \min_{a \in A} Q_{\theta^-}(s_{k+1}, a) - Q_\theta(a_k, a_k) \right]^2, \tag{2.2}$$

where θ are the parameters to learn for the deep neural network, and θ^- denotes the parameters of the target network that is updated periodically.

C51, short for Categorical DQN [4], is a variant of the Deep Q-Network (DQN) algorithm designed to enhance the representation and learning of the action value function. Instead of estimating $Q(s,a)$ directly on every state-action pair, C51 models the return distribution as a probability distribution over a fixed set of discrete atoms, each representing a possible return value. The core idea is that rewards in reinforcement learning are inherently stochastic and modeling their distribution provides more detailed information than a single expected value. In particular, the algorithm assigns probabilities to these discrete atoms, which enables effective learning on the distribution of Q-values, rather than their expected value.

2.2.2 Policy Optimization

The Policy Optimization approach explicitly represents the policy as a (stochastic) mapping $\pi_\theta(a \mid s)$, where we learn θ to fit the policy. These parameters are optimized either by directly performing a gradient ascent or by indirectly maximizing local approximations. Unlike Q-learning, policy optimization is usually conducted on-policy, meaning updates are made exclusively using data gathered while following the current version of the policy.

The fundamental advantage of policy gradient methods lies in their ability to directly optimize the policy without relying on state-value or action-value functions, making them particularly suitable for high-dimensional or continuous actions spaces [59]. In particular, these methods focus on directly learning the policy through a parameter vector θ, which governs the probability of selecting a specific action given a particular state:

$$\pi_\theta(a \mid s) = P(a_t = a \mid s_t = s, \theta). \tag{2.3}$$

Let $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right]$ be the additive reward over a trajectory τ simulated by policy π_θ, the gradient term $J(\theta)$ can be solved by the policy gradient theorem [58]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(u_t \mid s_t) G_t\right], \tag{2.4}$$

where G_t is the expected outcome estimated from time t and $\nabla_\theta \log \pi_\theta(u_t \mid s_t)$ is the log-likelihood gradient of the policy. This formula is the foundation of optimizing θ in such methods.

In actor-critic methods like A2C/A3C [32], an advantage function $A(s_t, a_t)$ is used to replace G_t, and is defined as:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t), \tag{2.5}$$

where $Q(s_t, a_t)$ is the Q-function given the current state and action. The state value function $V(s_t)$ is learned using parameter ϕ as: $V_\phi(s) \approx \mathbb{E}_{\pi_\theta}\left[G_t \mid s_t = s\right]$. The critic approximates $V_\phi(s)$ to assist the actor in calculating the advantage $A(s_t, a_t)$ for gradient updates,

with a typical loss function:

$$L(\theta) = -\mathbb{E}_{\tau\sim\pi_\theta}\left[\log \pi_\theta(a_t \mid s_t) A(s_t, a_t)\right]. \tag{2.6}$$

Proximal Policy Optimization (PPO) [46] algorithms focus on maximizing a special type of objective function that provides a pessimistic/conservative estimate of the policy performance $J(\pi_\theta)$ following an update. It could improve the stability and efficiency of training, preventing excessively large updates that could destabilize learning, which is achieved through mechanisms such as a clipped objective function or a KL-divergence penalty. It aims to balance simplicity, efficiency, and performance, which can be one of the most widely adopted RL algorithms in various domains, including robotic control, video games, and simulated physics, establishing it as a benchmark algorithm in reinforcement learning.

2.2.3 Bridging the two approaches

As policy optimization methods directly optimize the desired objectives, they typically result in greater stability and reliability during training. On the other hand, Q-learning methods train a neural network predictor to fit Q_θ and indirectly optimize agent performance. This indirect nature could fail due to the instability in the approximation of the function, bootstrapping [58], and off-policy data, making Q-learning methods less stable compared to policy optimization approaches. However, these two approaches are not mutually exclusive; there exists a range of algorithms that lie between these two extremes that balance the strengths and weaknesses of both approaches.

Deep Deterministic Policy Gradient (DDPG) [27] is an algorithm that simultaneously learns a deterministic policy and a Q-function, using each to improve the performance of the other. In particular, it borrows and combines the advantages of policy gradient and Q-learning methods to train a deterministic policy in continuous action spaces in an actor-critic fashion; the policy network $\mu_\theta(s)$ that outputs deterministic actions is used as the actor, critic is a Q-value network $Q_\phi(s,a)$ that evaluates the potential of actions. The training of critics is similar to Q-learning, while the actor is updated via maximizing the critic's Q-value via policy gradient. It also employs target networks and experience replay that are used in Q-learning for both the actor and the critic, which could stabilize

training by minimizing oscillations.

Soft Actor-Critic (SAC) [19] built upon DDPG to improve exploration, stability, and robustness. With the use of stochastic policies that output a distribution over actions rather than deterministic actions as in DDPG, it enables better exploration and learning in complex environments. Also, the key innovation is an entropy regularization term added to the policy optimization objective that could encourage exploration by maximizing the agent's robustness. In particular, the entropy term $H(\pi_\theta) = -\log \pi_\theta(a \mid s)$ includes a parameter that is capable of balancing exploration (in high potential regions) and exploitation (in high immediate reward). The policy is trained to maximize an entropy-augmented objective:

$$J_{\text{policy}}(\theta) = \mathbb{E}_{s\sim D, a\sim\pi_\theta}\left[\alpha \log \pi_\theta(a \mid s) - Q(s,a)\right]. \tag{2.7}$$

With better exploration and improved robustness in diverse environments, it is generally more effective for complex environments with high uncertainty and can score higher than DDPG on standard benchmarks.

2.2.4 Multi-Agent Reinforcement Learning

Many MARL algorithms are straightforward extensions of single-agent RL algorithms. Comprehensive reviews of various MARL algorithms can be found in [1] and [64]. In this section, we present two fundamental MARL algorithms closely related to the model-free RL methods discussed in earlier sections.

Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [28] is the multi-agent version of the DDPG framework that can address some unique challenges in MARL, such as non-stationary and high variance, while enabling agents to coordinate effectively in multi-agent environments. One major challenge in MARL is non-stationary for each agent. This is due to the constantly changing dynamics from other agents updating their policies, making the environment appear unstable and inconsistent for each agent, so Q-learning based methods that rely on experience replay struggle in such settings because environments are typically assumed to be stationary during training.

MADDPG tackles these challenges by utilizing an actor-critic framework that contains information on the policies of other agents during training, which allows MADDPG to handle the multi-agent environment effectively. In particular, each agent has an individual actor network that learns the policy privately based on its observation, and the outcome of the joint actions is evaluated using a centralized critic network accessible to all agents during training. It uses global state information and encourages agents to learn coordination behavior during training. Such a paradigm is often referred to as the Centralized Training and Decentralized Execution (CTDE) framework in MARL: During training, a central critic can receive the actual state signal and all agents' actions, providing more informative gradients for policy updates. However, during execution, only the actors are used and agents must operate independently based on their local information.

QMIX [45] is a Q-value based MARL algorithm that takes advantage of the concept of value decomposition in cooperative multi-agent settings. Specifically, a global Q-value $Q_{tot}(s,a)$ representing the total return of joint actions of all agents in the current state, is factorized into single and per agent Q-values $Q_i(s,a_i)$. This decomposition also adopts the CTDE framework, making QMIX highly effective for multi-agent environments. This algorithm assumes that the global Q-value can be expressed as a monotonic decomposition function of individual Q-value functions (i.e., $\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \forall i$), which ensures that an increase in any local Q-value will not lead to a decrease in the global Q-value, as the necessary condition to allow decentralized policies in a cooperative setting. It also introduces a mixing neural network that uses the local and global Q-values, and a key feature is that the network is parameterized by state-independent weights and biases, which allows the global Q-value to adapt to different states while maintaining monotonicity constraint. In the centralized training stage, the mixing network utilizes global state information to compute global Q-value effectively and ensures that agents can learn coordinated behaviors to optimize the shared global Q-value. After training, agents can operate independently by choosing their actions based on their local Q-values, without the need to access the global state or joint actions.

2.3 Rollout, off-line training and on-line play

As mentioned in the previous chapter, this thesis concentrates primarily on scenarios where the models are provided, namely planning methods. So far, the model-free methods reviewed have demonstrated excellence in many aspects, and the model-based methods have received less attention from researchers due to the availability of an accurate model and the complexity of learning a model. Nevertheless, the motivation of this thesis is a powerful framework named "off-line training, on-line play".

Proposed in [10] as a research monograph on RL and chess programs like AlphaZero [49], off-line training refers to the process in which the algorithm learns to evaluate chess positions and determine how to move towards advantageous ones. This training process uses a base chess player, which is iteratively refined using self-play, a conventional technique in RL, with the objective of developing a policy that guides the algorithm in making effective decisions. On-line play is the real-time application of the trained policy to generate moves against a live opponent, whether a human or a computer. It is a dynamic process that utilizes the knowledge gained from off-line training but enhances it with real-time computations, such as lookahead search and rollout simulations. An important observation about AlphaZero is that its on-line play policy significantly outperforms its off-line trained policy, despite the extensive training effort invested in the off-line phase.

According to the author, a key insight into AlphaZero's success is that *on-line play significantly enhances performance*, even when the off-line policy is extensively trained. This improvement could arise from the use of a few on-line planning methods, including lookahead optimization, rollout simulation, and terminal cost approximations. In particular, the lookahead optimization refers to a one-step or multi-step lookahead search that is performed to evaluate possible future positions, which involves simulating potential moves via rollout simulation based on an off-line trained base policy. For long horizon, or discounted infinite horizon problems, a terminal cost approximation can be used to terminate the rollout simulation and return the expected outcome. After evaluating different moves, the algorithm chooses the one with the best outcome and uses it as the actual move in the game. After that, the algorithm will repeat the previous steps in all subsequent states or positions, which is similar to receding horizon approach/model predictive control. This combination allows the on-line play policy to adapt to specific

game scenarios in real-time, making it far more effective than relying solely on the off-line policy.

In addition, the author provided an abstract concept of the on-line play policy's resemblance to Newton's method in optimization. It is emphasized in the book that the improvement in performance is often evident and could be a result of the super-linear convergence ability of Newton's step. In particular, the author proposed that approximating the value space through one-step lookahead minimization is equivalent to taking a step of Newton's method to solve Bellman's equation. Additionally, the author argued that the starting point for the step is derived from the off-line training policy and can be further improved through longer lookahead minimization and on-line rollout.

Based on his theories and insights, we conclude that off-line efforts alone might not be enough, especially in complex and dynamic environments, and an on-line approximation in the value space, as seen in AlphaZero could dramatically improve the results. This motivates our research on an on-line planning algorithm in a decentralized multi-agent setting. As stated earlier, in MARL with limited communications, agents rely solely on off-line trained policy in a centralized fashion like MADDPG and QMIX. While an off-line lookahead maximization is allowed during the centralized training, the on-line play policy like the one used in AlphaZero is technically unachievable during execution under a decentralized and cooperative scenario.

2.4 Decentralized Partially Observable Markov Decision Process

Decentralized POMDPs, or Dec-POMDPs, are an extension of POMDPs to multi-agent systems. In Dec-POMDPs, multiple agents operate based on the local operation history to achieve a joint objective, such as maximizing cumulative rewards over time. Training agents in this model is challenging due to factors such as limited communication, partial observability, and the need to make independent decisions without accessing information from other agents.

The Dec-POMDP framework builds upon concepts from MDPs, POMDPs, and multi-agent systems. It was formally introduced in [6], establishing its complexity class as

NEXP-complete and defining it as a decentralized control problem. Early algorithms, such as Joint Equilibrium-based Search for Policies (JESP) [34], aimed to solve joint policies by iteratively enforcing equilibrium conditions. However, as the size of the problem increases, the exponential expansion of the joint policy space remains the primary barrier to scalability.

Memory-Bounded Dynamic Programming (MBDP) [47] is an approximate method designed to address the scalability issue in Dec-POMDPs by limiting the size of policy space that agents can search, ensuring computational feasibility for relatively large size problems. Instead of considering all possible policies, a finite state controller (FSC) that has small, bounded memory states is used to express the policy of each agent. The FSC maps an agent's local observations to actions based on its memory state that evolves over time, and it employs a backup operation that computes approximate value functions for the memory-bounded policies. Additionally, the algorithm iteratively refines the policies for a fixed memory size. If the performance of the policies is undesirable, the memory bound can be increased to allow for more expressive policies. Policy Iteration (PI) for Dec-MDPs [5] is an adaptation of the classic policy iteration in MDPs and POMDPs to decentralized control settings. Although it can iteratively enhance the policy by alternating between policy evaluation and policy improvement, in decentralized settings, policies must account for each agent's limited observations, partial information, and decision-making with communications. The adaptations of PI for Dec-POMDPs include: memory-bounded policy iteration that combines PI with memory constraints similar to MBDP to reduce computational costs; decentralized policy iteration with belief compression to reduce the policy space by compressing belief states/histories, enabling faster convergence; heuristic policy iteration that use heuristics to guide policy improvement.

Monte Carlo Tree Search (MCTS) methods have emerged as practical approaches to solve Dec-POMDPs, particularly for large-scale problems. By combining tree-based search with Monte Carlo sampling, these methods efficiently approximate the value function and policy without requiring an exhaustive enumeration of all possibilities. They could be well-suited for Dec-POMDP as simulation-based sampling is used to explore joint action-observation spaces, which can handle partial observability through belief/particle sampling, and balance exploration and exploitation dynamically during policy construction. Dec-MCTS [2] also generalizes the standard MCTS algorithm to decentralized mutli-agent settings. As agents must make decisions independently without

direct knowledge of the environment state and other agents' information, Dec-MCTS tackles this challenge by building decentralized/local search trees for each agent. Although decisions are decentralized, agents can implicitly coordinate by sampling joint trajectories that maximize the joint cumulative reward.

Recent advances in MARL have also enabled scalable approximation techniques for Dec-POMDPs by utilizing deep learning and decentralized policies, including MADDPG [28] and QMIX [45] which are discussed previously. Most algorithms leverage the CTDE paradigm, namely, the agents use centralized information during training to optimize policies, but at runtime, they act solely on local information. Some other algorithms include: Value Decomposition Networks (VDN) [57] that can decompose the joint Q-function into the linear combination of per-agent Q-functions. Unlike the monotonic decomposition in QMIX, VDN uses a linear value decomposition that simplifies the learning process. Counterfactual Multi-Agent Policy Gradients (COMA) [17] uses counterfactual baselines to solve the problem of credit assignment in multiple cooperative agents. The joint Q-value for all agents is evaluated by a centralized critic, while counterfactual baselines can measure the individual contribution of each agent's action to the total reward. Factorized Multi-Agent Centralized Policy Gradients (FACMAC) [42] extend MADDPG to handle cooperative tasks more effectively by introducing a factorized critic, which decomposes the joint Q function into agent-specific components in a way similar to QMIX and VDN, while using policy gradients to optimize continuous actions. Multi-Agent Variational Exploration (MAVEN) [30] also builds on QMIX and VDN by introducing a latent space to model exploration strategies between agents, which could address the difficulty of coordinated exploration in Dec-POMDP, where agents must explore joint action spaces effectively.

2.5 Common Information Approach

The common information approach [35] has emerged as a powerful framework in decentralized stochastic control, addressing coordination challenges in multi-agent systems where agents operate with asymmetric information and limited or delayed communication. This approach is particularly applicable in areas like multi-robot coordination, networked systems, and distributed sensor networks, where effective coordination is necessary but agents are unable to directly share their private observations.

The common information approach was initially proposed to address the inherent complexities of decentralized decision-making. In such settings, agents possess private information and observations that are not directly accessible to others. To address this, the framework introduces a "virtual coordinator" that utilizes the "common information" observable by all agents. This coordinator reformulates the decentralized problem into a centralized one by determining mappings that guide agents' operation based on their local private information only. In this way, it enables agents to coordinate decisions without the need for direct sharing of private observations. The fundamental work in this field is [35], which introduced the use of common information to solve decentralized stochastic control problems when partial sharing of the agent's history is allowed. They demonstrated how this approach can convert a decentralized control problem into an equivalent centralized one where the virtual "coordinator" prescribes actions to agents. The coordinator operates within a high-dimensional action space of mappings, which encode strategies that assign mappings to map each agent's private information to an action. This reformulation allows the use of centralized techniques to approximate solutions, facilitating improved coordination in settings with information asymmetry.

Building on this foundation, studies in [40] extended the common information approach to decision making problems with asymmetric information. The author introduced methods for constructing beliefs over agents' private states, leveraging common information to dynamically update these beliefs. By maintaining shared beliefs, agents can more accurately predict others' behavior and adapt their actions accordingly. This work highlights how common information can be used to bridge the gap between decentralized observations and coordinated decisions, even when communication delays or partial observability hinder direct synchronization.

While the common information approach has demonstrated notable benefits, its implementation in large-scale multi-agent systems presents several challenges: The coordinator's action space grows significantly as the number of agents and observations increases, leading to computational intractability. Techniques such as policy pruning, hierarchical decomposition, and approximate solutions (e.g. hybrid policies) are required to address this issue. Constructing beliefs over agents' private histories becomes increasingly complex as the system dynamics evolves. The dynamic belief update methods [40] offer a starting point, but scalability remains an open problem. In open environments

where agents are allowed to join or leave, common information must dynamically adapt to accommodate changes in group composition and shared knowledge. RL methods that incorporate on-line planning or meta-learning offer promising directions to address this. Since the coordinator works with mappings, the dimensionality of the action space can explode in large-scale problems. Point-based value iteration and multi-agent rollout algorithms have been explored to approximate the solution space effectively.

This approach remains a robust and versatile framework for addressing coordination problems in decentralized multi-agent systems. Using shared information and reformulating decentralized problems as centralized ones, this approach enables agents to achieve effective coordination while respecting information asymmetry and communication constraints.

2.6 Applications: Control of Microgrid with MARL

Numerous studies have explored the application of MARL in microgrid systems. In [25], an optimal control strategy for a grid-connected microgrid was proposed, featuring a multi-agent system that minimizes data manipulation and exchange. Each agent operates with limited information for its own bus. The study used MARL to address the "curse of dimensionality" and reduce electricity costs in the microgrid. In [16], a framework for the scheduling of residential microgrid energy was introduced that incorporates vehicle-to-grid systems to enhance autonomy and fairness in the microgrid market. The authors developed an MARL algorithm based on an improved equilibrium selection that negotiates with agents to compute equilibrium solutions and selects the optimal one based on the highest average reward. The authors of [36] proposed a MARL-based model to address post-disaster resilience in island microgrids, converting the objective of powering as many loads as possible after a disaster into a policy optimization problem that seeks to maximize the cumulative utility of the microgrid during the post-disaster period.

In [43], a power control paradigm was developed that integrates edge computing with MARL. Edge nodes can detect network states and control the power electronics converter to enable rapid response and local autonomy. The study also addressed optimal power flow adjustment, with a distributed MARL algorithm that provides in-

telligent, system-aware decisions. In [33], an energy scheduling scheme was proposed for microgrid-powered multi-access edge computing, with the aim of minimizing the gap between estimated supply and demand. The authors formulated a risk-aware scheduling problem as a multi-agent stochastic game with a joint policy, solving it using a MARL algorithm. The authors of [65] introduced a MARL scheme in a partially observable and multi-agent environment to manage microgrid energy with communication failures. They developed a multi-agent Bayesian deep RL algorithm to address communication issues and select optimal joint actions in a decentralized manner. In [26], a distributed load frequency control framework was designed to facilitate collaboration between controllers and distributors in microgrids. The study proposed a novel distributed deep RL algorithm that offers enhanced adaptability, robustness, and reduced decision-making time.

2.7 Applications: Networked Microgrid with MARL

In the context of networked microgrid systems, the authors of [13] studied external peer-to-peer (p2p) energy trading and internal energy conversion within interconnected multi-energy microgrids spanning residential, commercial and industrial sectors. Given the complexity of these high-dimensional and uncertain decision-making problems, they proposed a MARL approach that combines the multi-agent actor-critic algorithm. In [37], a transactive energy trading platform was developed to incorporate energy storage systems into the energy management of microgrids. This platform facilitates participation at the local, communal, and global levels in the interconnected microgrid energy market. The authors introduced a MARL algorithm, called simulated annealing-based Q-learning, to optimize the bidding strategies of energy storage systems, which greatly improves profits from trading in the energy market. In addition, a coordinated energy scheduling technique for distributed multi-microgrid systems involving central energy management agents and coordination control agents was proposed in [56]. The study introduced a multi-step planning framework aimed at minimizing the internal operational costs of microgrids while maximizing the benefit of the entire multi-microgrid system. This problem was addressed using a specific particle swarm optimization algorithm that was implemented iteratively among all agents.

2.8 Applications: Frequency Regulation in Microgrids

In [53], a hierarchical framework was proposed for microgrids with clusters of distributed energy resources to engage in the frequency regulation market. The framework consists of two layers. The top layer is responsible for comprising aggregators that independently communicate to disaggregate regulation signals from the market operator, while the bottom layer aggregates all distributed energy resources within the microgrids. These resources are controlled and adjusted to ensure that the output power matches the one requested by the top layer. A distributed algorithm was also developed to address load uncertainties and power flow limitations. In [55], a per-user-based strategy was introduced to enhance the profitability of battery storage units in markets such as energy trading and ancillary services. The authors proposed a rental strategy and developed a mathematical model for the commercialization of battery equipment within distributed electricity networks, ensuring both the optimality of the company and the customer. Tested with data from the Australian electricity market, they also proposed a rental pricing algorithm that incorporates the profit gained by batteries participating in the electricity market to support investment and operation planning.

In [63], a capacity bidding strategy was proposed in wind storage systems, using robust predictive control of models, to maximize revenue obtained from the frequency regulation market. The authors developed a mixed integer programming algorithm that considers wind power forecast uncertainty, energy market prices, and regulation market prices, allowing the system to optimize energy trading by leveraging low-cost energy. In [39], a coordinated control strategy was introduced for a virtual power plant to provide frequency regulation services. The virtual power plant, which consists of distributed battery storage and other units such as water heaters, employs an intelligent supervisory coordinator to address frequency deviations caused by long-time delays. The authors of [20] proposed a decision supporter for electric vehicle aggregators to participate in both day-ahead and real-time markets to regulate frequency and energy trading. They developed a mathematical model of the behavior of electric vehicles within a smart grid environment, incorporating a risk measure called conditional value-at-risk to produce offers that balance conservatism and optimism.

2.9 Gaps in the Literature

It can be perceived that researchers have extensively explored the application of MARL in microgrids, covering areas such as control, scheduling, optimization, stability, reliability, energy trading, and resilience. According to our review, many algorithms consider centralized and fully observable systems with perfect and instantaneous communication. Also, the applications in the networked microgrid system do not receive much attention, with only a few papers using the multi-agent RL technique. This is not surprising given the inherent "curse of dimensionality" in dynamic programming/RL, which would be worsened in partially observable and decentralized environments. Unlike those benchmark problems tested in the RL papers, controlling a power grid would involve much larger state/action/observation spaces, and therefore solving optimal policies in complex environments could be intractable. In addition, in small-scale microgrids (i.e., residential communities, suburbs, and universities with renewable power supply), the centralized control/communication paradigm could be approximately achievable within a certain physical range, while communication and operation delays would be inevitable when cooperating with a network of microgrids across different regions. As a result, due to scalability issues and a lack of the beneficial "on-line play" policies, designing a fully decentralized power system following the standard Dec-POMDP model would be undesirable with many stochastic/noisy signals in the real world. Our decentralized multi-agent planning algorithm could be a potential starting point to fill this gap, by adopting a means of implicit communication to enable on-line planning in decentralized environments.

Networked microgrid systems also raise concerns that are beyond the scope of standard MARL algorithms. A reasonable assumption when connecting multiple microgrids is that there have already been some control/learning algorithms designed to adapt to the specifications (i.e., stochastic renewable generation, power demand, and electricity prices) in each system. In other words, each already has some suboptimal controllers/policies corresponding to solving specific POMDP or Dec-POMDP models. At the network level, this somewhat creates a meta-level decision/cooperation problem that could be approximately modeled by adopting a larger POMDP/Dec-POMDP, but the existing algorithms in each microgrid could be unusable due to the change in factors such as state/observation space, reward functions, and information sharing patterns. Ideally, this would require a group of agents to cooperate with another or multiple groups of agents with little/without prior

coordination, using their existing built-in policies trained in their specific environment. Therefore, utilizing MARL alone is insufficient for integrating a network of microgrids and would require further knowledge from research fields such as domain adaptation, ad hoc teamwork, and transfer learning.

Although it is beyond the scope of this thesis, the importance of prosumers' trust in smart grid applications is generally neglected. There are some agents that can be directly affected by prosumers' preferences and activities, such as controllable loads in demand response and electric vehicles. This somewhat creates opportunities to utilize approaches in RL with human feedback (RLHF), and we believe that incorporating prosumers' trust would be essential to design products tailored to prosumers' need, especially when their vehicles and house appliances are handled by an AI system. This could be a crucial factor when designing the next generation power system, and studies of RLHF agents that are capable of incorporating user trust into the decision-making process of the system, similar to the user trust feedback loop framework described in [66] could be a promising research direction.

2.10 Summary

In this chapter, we conducted a thorough review of the basic framework in RL, focusing on both model-free and model-based methods, and extensions to decentralized multi-agent environments. The main contribution of this thesis is on enabling planning during execution in multi-agent environments with limited/indirect communication, and we reviewed the essential knowledge required to achieve it, including the design paradigm of "off-line training, on-line play", Dec-POMDP frameworks, and common information approaches. Modern applications of MARL in smart grids are also reviewed, and we listed some of the potential gaps in the current literature.

CHAPTER 3

DECENTRALIZED MULTI AGENT PLANNING ALGORITHM

3.1 Problem Formulation

This section provides a detailed explanation of the decentralized multi-agent planning algorithm, beginning with the Dec-POMDP model [38], which is characterized by the tuple $(I, S, A, M, S, O, b_0, \mathbb{P}, r, \beta)$, where

- I represents the agent set, S is the state space.
- $A = \prod_{i \in I} A^i$, where A^i is the action space of agent i.
- $O = \prod_{i \in I} O^i$ is the set of observations for agent i.
- $M = \prod_{i \in I} M^i$, where M^i is the set of the private information in agent i, and is a sequence of A^i and O^i.
- b_0 is the initial distribution of belief state.
- $\mathbb{P} : S \times M \times A$ is the combined transition and observation probabilities.
- $r : S \times A \to \mathbb{R}$ is the immediate reward function.
- $\beta \in (0, 1)$ is the discount factor.

The common information approach [35] offers a comprehensive framework for decentralized stochastic control, relying on the assumption that agents share a subset of

their past data (observations and actions) via a shared memory, with all agents retaining perfect recall of this common information. Within the Dec-POMDP model, these shared data are viewed as common observations. The core idea of the approach is to reformulate the decentralized problem into an equivalent centralized one from the perspective of a virtual "coordinator" that has access to the common information. This "coordinator" assigns "prescriptions," which map each agent's local information to its corresponding action [35]. Since the "coordinator" cannot directly access each agent's private information, the system remains decentralized.

The authors in [35] provided some cases of information structure that fit the common information approach, including delayed sharing, delayed state sharing, periodic sharing, control sharing, and no shared memory. One perspective to interpret this approach is to consider a cooperative decentralized multi-agent card game such as Hanabi [3]. According to [18], it is a game in which players are required to play a legal sequence of cards and aids from other players are allowed to give hints on which cards are of a specific rank or color. The cards openly played on the table represent the common information as they include shared details from all players, while the cards held by each player constitute their private information.

Consequently, using the common information approach, the Dec-POMDP model can be reformulated into a coordinator's POMDP as follows:

- $\bar{O}$ is a common observation available to all agents.
- $\bar{S} = S \times M$ is the state space augmented by private information.
- $\bar{A} = \prod_{i \in I} \bar{A}^i$ is the action space for the coordinator, where $\bar{A}^i$ is the set of mappings(prescriptions) from M^i to A^i.
- $\bar{\mathbb{P}} : \bar{S} \times \bar{A}$ is the combined transition probability.
- $\bar{r} : \bar{S} \times \bar{A} \to \mathbb{R}$ is the immediate reward function.

The transition probability and reward function are defined as:

$$\begin{aligned} \bar{\mathbb{P}}(\bar{s}', o \mid \bar{s}, \gamma) &= \mathbb{P}(\bar{s}', o \mid \bar{s}, a), \qquad \forall \bar{s}, \bar{s}' \in \bar{S}, \bar{o} \in \bar{O}, \gamma \in \bar{A}, \\ \bar{r}(\bar{s}, \gamma) &= r(s, a), \qquad \forall \bar{s} \in \bar{S}, \gamma \in \bar{A}, \end{aligned} \tag{3.1}$$

where $\bar{s} = (s, m)$, and $a^i = \gamma^i(m^i)_{i \in I}$. In the coordinator POMDP, the "prescriptions" γ and the common information $\bar{O}$ are defined as the actions and the observations, respectively. Also, a belief over the augmented state $\bar{S}$ serves as the information state for the CPOMDP. The proof of 3.1 can be found in [35], see stages 2 to 4 in Section 3 of the paper, and Appendix A for more detail.

Remark 3.1: It should be pointed out that the space of prescriptions is prohibitively enormous and grows with the number of planning horizons and agents, even for relatively small problems. This is due to the fact that $|\bar{A}| = \prod_i |A^i|^{|M^i|}$, and in this chapter, the agent-by-agent approach [8] is used to tackle this curse of high dimensionality. In addition, for planning in infinite horizon discount problems, the private information M could be an infinite sequence of observations and actions. An alternative approach to handle this is to use a compression of private information, which could be (approximately) sufficient for decision making, including an approximate information state [54] and sufficient private information [61]. However, in the CPOMDP model [60], the private information M is assumed to be a fixed function of the state $\bar{A}$, which is reasonable as the state in CPOMDP is a combined and enhanced state of both.

3.2 Methodology

In this section, we present an algorithm for solving the coordinator's POMDP from the perspective of an "on-line play policy." The algorithm integrates techniques such as point-based value iteration, heuristic search value iteration, one-step lookahead rollout simulation, and Monte Carlo tree search. To simplify the explanation, we will first outline the algorithm using the standard notation of a single-agent POMDP and later demonstrate its extension to a coordinator POMDP involving multiple agents in the subsequent section.

3.2.1 Point-Based POMDP Solvers

As a belief state serves as a sufficient statistic to encapsulate the information vector of a POMDP (see Appendix A1 for proof), a POMDP can be reformulated as a perfect state

information problem defined over the belief state space.

Consider a system with n states labeled $i = 1, 2, \cdots, n$. The belief state is represented by a conditional probability vector $b = (b(1), b(2), \cdots, b(n))$, where $b(i)$ denotes the conditional probability of the state being i, given the action-observation history. It can be updated sequentially using a belief update formula based on Bayes' Theorem (see Appendix A2 for details). The optimal value function $V^*(b)$ is the solution of the following Bellman equation:

$$V^*(b) = \max_{a \in A} r(b,a) + \beta \sum_{o \in O} \mathbb{P}(o \mid b,a) V^*(b'). \tag{3.2}$$

Despite the belief space being infinite, early work [52] showed that the optimal value function for a finite-horizon POMDP can be expressed using a finite set of vectors. Similarly, for the discounted infinite-horizon case, the optimal value function can be arbitrarily closely approximated by a finite set of vectors, commonly referred to as alpha vectors. In particular, alpha vectors have the same size of states, and each vector corresponds to a linear segment of the value function in the belief space and is associated with a particular action/policy [48]. Let $V = \{\alpha_1, \cdots, \alpha_{|V|}\}$ represent the set of alpha vectors defined over the belief space. The value of a specific belief is calculated as follows:

$$V(b) = \max_{\alpha \in V} b \cdot \alpha, \tag{3.3}$$

where $b \cdot \alpha = \sum_{s \in S} b(s) \cdot \alpha(s)$ is the dot product between the alpha vector and the belief state.

The point-based value iteration algorithm for solving large POMDPs operates by selecting a finite set of belief points and maintaining alpha vectors that are optimal for at least one belief in this set. This method balances a trade-off: reducing the size of the alpha vectors representation to prevent exponential growth comes at the expense of decreased accuracy in approximating the optimal value function.

An effective strategy is to focus only on reachable belief points by maintaining a subset of belief points that are achievable under the (optimal) policy. Additionally, pruning alpha vectors that are not optimal for this subset of belief points has shown promising

results in tackling large POMDPs (see [24], [50], and [51]). Under this paradigm, the core step for updating new alpha vectors is called a backup operation, which uses the subset of sampled belief points to iteratively refine the value function. It is expressed as follows: Let the value function for the next belief be $V(b') = \max_\alpha \sum_{s'} b'(s')\alpha'(s')$, Substituting this into equation 3.2, we get:

$$V(b) = \max_{a \in A}\big(r(b,a) + \beta \sum_{o \in O} \mathbb{P}(o \mid b,a) \max_{\alpha' \in V} \sum_{s' \in S} b'(s')\alpha'(s')\big). \tag{3.4}$$

And using the belief update formula $b'(s') = \frac{\mathbb{P}(o|s',a)\sum_s \mathbb{P}(s'|s,a)b(s)}{\mathbb{P}(o|b,a)}$ (see Appendix A2), we obtain

(3.5)

$$\begin{aligned} V(b) &= \max_{a \in A}\big(\sum_{s \in S} b(s)r(s,a) + \beta \sum_{o \in O} \mathbb{P}(o \mid b,a) \max_{\alpha' \in V} \sum_{s' \in S} \frac{\mathbb{P}(o \mid s',a)\sum_s \mathbb{P}(s' \mid s,a)b(s)}{\mathbb{P}(o \mid b,a)}\alpha'(s')\big) \\ &= \max_{a \in A}\big(\sum_{s \in S} b(s)r(s,a) + \beta \sum_{o \in O} \max_{\alpha' \in V} \sum_{s' \in S} \mathbb{P}(o \mid s',a)\sum_s \mathbb{P}(s' \mid s,a)b(s)\alpha'(s')\big) \\ &= \max_{a \in A} \sum_{s \in S} b(s)\big(r(s,a) + \beta \sum_{o \in O} \sum_{s' \in S} \mathbb{P}(o \mid s',a)\mathbb{P}(s' \mid s,a) \max_{\alpha' \in V} \alpha'(s')\big). \end{aligned}$$

Recall that $V(b) = \max_\alpha b \cdot \alpha$, then the term inside the bracket represents the alpha vector update rule:

$$\alpha_a(s) = r(s,a) + \beta \sum_{o \in O} \sum_{s' \in S} \mathbb{P}(o \mid s',a)\mathbb{P}(s' \mid s,a) \max_{\alpha' \in V} \alpha'(s'), \tag{3.6}$$

where $\max_{\alpha' \in V} \alpha'(s')$ is the best alpha vector for the next state state s' chosen from the current set of vectors V. Thus, for a given belief point b,a backup operation can be defined to generate a new alpha vector using equation 3.6, expressed as:

$$backup(V,b) = \arg \max_{a \in A, \alpha \in V} b \cdot \alpha_a. \tag{3.7}$$

3.2.2 Heuristic Search Value Iteration

The HSVI [50], HSVI2 [51], and SARSOP [24] algorithms are designed based on the point-based algorithm. In addition to the alpha vectors representing a lower bound

as a conservative estimate of the optimal value function V^*, they introduced a belief point-based upper bound as an optimistic estimate of V^*. The main objective in these algorithms is to minimize the gap between the upper and lower bounds among some belief points, which can be done by the backup operation to update the alpha vectors. Also, they use different forward exploration heuristics to sample belief points that are reachable [50, 51], or reachable under an optimal policy [24].

In particular, the lower bound in HSVI is a uniform and pessimistic policy of the form "always execute action a", and the worst-case outcome for the long-term reward of executing this policy is:

$$\underline{r}_a = \sum_{t=0}^{\infty} \beta^t \min_s r(s,a) = \frac{\min_s r(s,a)}{1-\beta}. \tag{3.8}$$

The tightest of these bound is chosen by maximizing:

$$\underline{r} = \max_a \underline{r}_a, \tag{3.9}$$

and the vector set for the initial lower bound contains a single alpha vector such that $\alpha(s) = \underline{r}$.

Conversely, the upper bound is initialized by solving the corresponding fully observable MDP, providing an optimistic estimate of the value. Assuming that the agent knows the exact state s at every step, we can calculate the optimal $V^{MDP}(s)$ for every state s, and given a belief point $b(s)$, an upper bound value can be calculated as:

$$\overline{V}(b) = \sum_s b(s) V^{MDP}(s), \tag{3.10}$$

which represent the best possible value assuming full observability.

The belief points sampling strategy used in HSVI is a forward exploration heuristic, which is described as follows: First, define an interval Q function $\hat{Q}$ as $\hat{Q}(b,a) = [Q^{\underline{V}}(b,a), Q^{\overline{V}}(b,a)]$, which represents the uncertainty of estimating the optimal value function in b by choosing a. Then among all $Q^{\underline{V}}$ intervals, choose an action that generates

the largest upper bound by:

$$\hat{a} = \arg\max_{a} Q^{\overline{V}}(b,a), \tag{3.11}$$

which could be a good way to guarantee convergence, since repeatedly choosing an $\hat{a}$ that is suboptimal will eventually drop the $\hat{a}$ upper bound below the upper bound of another action [50]. This is similar to the IE-MAX heuristic introduced in [23].

To find the next belief point to update, an observation $\hat{o}$ is chosen from the child nodes of action $\hat{a}$ with the maximum gap, allowing the algorithm to target belief points where the current gap is unsatisfactory, namely the estimation has the greatest uncertainty. Consider the relationship between $\hat{Q}(b,\hat{a})$ and the bounds at the child belief nodes $b' = \tau(b,\hat{a},o)$ that are reachable under different observations, which is:

$$width(\hat{Q}(b,\hat{a})) = \beta \sum_{o} \mathbb{P}(o \mid b,\hat{a}) width(\hat{V}(\tau(b,\hat{a},o))). \tag{3.12}$$

Given a desired gap ϵ between the upper and lower bounds, the exploration termination criterion is defined as $width(\hat{V}(b) \leq \epsilon\beta^{-t}$, as the uncertainty at a belief node b after an update is at most β times a weighted average of its child nodes. In other words, the requirements on deeper nodes have successively looser requirements on uncertainty. Based on these facts, define the excess uncertainty at a belief node b with depth t in the search tree as:

$$excess(b,t) = width(\hat{V}(b)) - \epsilon\beta^{-t}. \tag{3.13}$$

It can be perceived that if all child nodes from b have negative excess uncertainty, the belief point b will also have negative excess uncertainty after an update, implying the desired convergence to within ϵ. As a result, the forward exploration heuristic can be designed to focus on child nodes that contribute most to the excess uncertainty to ensure early termination, namely the next belief to explore is sampled from the observation that maximizes the excess gap. It will be:

$$\hat{o} = \arg\max_{o} \left[\mathbb{P}(o \mid b,\hat{a}) excess(\tau(b,\hat{a},o), t+1)\right]. \tag{3.14}$$

In summary, the objective of these algorithms is to minimize the gap between the convex hull formed by all sampled belief points (upper bound) and the lower envelope formed by the linear combination of alpha vectors at all sampled belief points (lower bound). The sampling strategy differs in these algorithms, as HSVI and HSVI2 focus on reachable belief points given an initial belief point, while SARSOP focuses on reachable belief points under an optimal policy.

3.2.3 One-Step Lookahead Optimization with Rollout Simulation

A suboptimal solution approach introduced in [10] [12] is based on an approximation via simulation in the value space and an optional cost approximation at the end of the rollout simulation. It is the "on-line play" policy stated earlier, with the basic idea of replacing the optimal value function V^* in the Bellman equation 3.2 with an estimated value $\tilde{V}_{rollout}$ to compute the policy $\tilde{\pi}$ with one-step lookahead maximization, which is:

$$\tilde{\pi} = \arg\max_{a \in A} \big[r(b,a) + \beta \sum_{o \in O} \mathbb{P}(o \mid b,a) \tilde{V}_{rollout}(b'(o,a),d) \big]. \tag{3.15}$$

.

d is the depth of the rollout simulation. If we use the alpha vectors as the elements of base policies, namely the direct control policy $\pi_{base}(b) = \max_{\alpha \in V, a \in A} b \cdot \alpha_a$, the approximated value of the value function $\tilde{V}_{rollout}$ via the rollout simulation will be:

(3.16)

$$\tilde{V}_{rollout}(b,d) = \begin{cases} \max b \cdot \alpha_a & \text{if } d = 0 \\ r(b,\pi_{base}(b)) + \beta \sum_o \mathbb{P}(o \mid b, \pi_{base}(b)) \tilde{V}_{rollout}(b'(o,\pi_{base}(b)), d-1) & \text{if } d < 0. \end{cases}$$

$d = 0$ indicates the rollout has reached the end of the simulation, and, for infinite discounted or problems with very long horizons, a terminal cost approximation can be used at the end of the rollout simulation.

The base policy can range from simple strategies, such as random or greedy policies, to complex policies extensively trained offline. As emphasized in [10], the effectiveness of an off-line trained policy can be greatly improved through the on-line value space

approximation, as well as incorporating long lookahead steps (involving optimization and/or rollout with the off-line policy) and terminal cost approximations obtained off-line. The author attributes this improvement to the observation that approximating the value space using one-step lookahead maximization is analogous to taking a Newton's method step for solving Bellman's equation, which is known for its superlinear convergence. A key takeaway from [10] is that off-line trained policies often fall short compared to their one-step or multi-step lookahead counterparts, as they lack the additional refinement achieved through the Newton step.

3.3 Our Algorithm: Decentralized Multi-Agent Planning

Our proposed algorithm is a decentralized multi-agent planning framework constructed on the basis of both approaches. We use point-based and heuristic search algorithms as the starting point for base policies that are needed in the rollout simulation and terminal cost approximations. The general framework of our algorithm is the policy iteration with repeated rollout for improvements, which can be considered as self-learning.

As the core idea of point-based methods is to linearly approximate POMDPs' value functions at selected/representative belief points using sets of alpha vectors, obtaining near-perfect approximating vectors for value functions at every belief point is intractable. Instead of solving the exact value function, we use these approximating features to guide the agents during execution as a means of "on-line play".

It is important to emphasize that the effectiveness of rollout simulation heavily depends on the quality of the base policy. As discussed in [11], a truncated rollout using a reasonably good (e.g., stable) base policy can achieve performance comparable to that of extended lookahead optimization, which requires significantly higher computational costs. This conclusion is supported by extensive computational experiments, starting with the rollout framework used in TD-Gammon [62].

As a result, instead of purely greedy or random policies, we developed a novel hybrid policy that is inspired by the "gap shrinking" mechanism in HSVI, which leverages approximating features including backup operation's lower bounds, IE-MAX heuristic's

uncertainty gaps, and optionally, entropy-based uncertainty on belief distribution.

3.3.1 Hybrid Base Policy

As stated previously in Sections 3.2.2 and 3.2.3, the standard direct control policy obtained from the backup operation (Equation 3.7) can serve as the lower bound in the heuristic value iteration by assuming a worst case/conservative form of policy execution (Equations 3.8 and 3.9). This backup-based policy can be considered as an exploitation-driven and greedy type of policy on the current lower bound, which focuses on maximizing the immediate rewards and approximations already encoded in the lower bound. It could be a baseline value that represents what we already know and how certain we are about the value function, using alpha vectors as means of approximation.

On the other hand, the IE-MAX-based policy used in heuristic search (see equations 3.10 and 3.11) is exploration driven, prioritizing regions of the belief space where the bounds have the most uncertainty. It could be perceived as an uncertainty measure that quantifies the potential improvement in approximating the value function. In other words, a rollout simulation that uses IE-MAX-based policy could simulate trajectories towards areas of the belief space with high potential of improving the value function approximation.

In an effort to strike a balance between exploitation (backup) and exploration (IE-MAX), we combine these two approaches to produce a base policy that can guide the rollout simulation to approximate a value function that represents both the certainty from backup and the uncertainty from IE-MAX. Given a belief point b, a hybrid base policy is given as:

$$\pi_{hybrid}(b) = \arg\max_{a}\big(\lambda \cdot \mathbb{E}[\underline{V}(b)] + (1-\lambda)\cdot \mathbb{E}[\Delta(b)]\big), \tag{3.17}$$

where $\lambda \in [0,1]$ is a balancing parameter between exploitation $\mathbb{E}[\underline{V}(b)]$ and exploration $\mathbb{E}[\Delta(b)]$. The exploitation term represents the expected lower bound value in successive beliefs, and the exploration term $\mathbb{E}[\Delta(b)] = (\overline{V}(b) - \underline{V}(b))$ represents the expected gap in successive beliefs. This hybrid policy is inspired by the TD-lambda learning algorithm [58]; the backup-based policy dominates as $\lambda \to 1$ to prioritize exploitation, and the

IE-MAX policy dominates as $\lambda \to 0$ to prioritize exploration.

Instead of a fixed λ, we can adopt a dynamic balancing of λ. Initially, the average gap across all belief points is large with the default upper and lower bounds used in HSVI, and a low λ can be used (i.e., below 0.5) to favor exploration. As the gaps between all beliefs gradually reduce as a result of convergence, we can increase the value of λ to favor exploitation. Therefore, we can define a relative gap-based uncertainty as follows:

$$\textbf{Uncertainty}(b) = \frac{\Delta(b)}{\overline{V}(b)} = \frac{\overline{V}(b) - \underline{V}(b)}{\overline{V}(b) + \epsilon}. \tag{3.18}$$

If the bounds are close, the value of $\textbf{Uncertainty}(b) \to 0$, indicating high confidence in the approximation of the value function in b; while $\textbf{Uncertainty}(b) \to 1$ indicates high uncertainty on the approximation;a small value of ϵ prevents the division by zero. Also, adding an entropy-based uncertainty to the belief distribution $b(s)$ could be helpful to reflect the inherent uncertainty of the belief distribution, such as $\textbf{Entropy}(b) = -\sum_s b(s) \log b(s)$. In other words, a high value of $\textbf{Entropy}(b)$ could indicate a relatively uniform distribution over the states, implying a large uncertainty. And if b is concentrated in a single state (low uncertainty), the value of $\textbf{Entropy}(b)$ should be low. Nevertheless, the computation for a logarithm of belief state could be costly in a multi-agent problem with many states, so we omit the entropy uncertainty and use only the relative gap uncertainty. As a result, the dynamic λ is defined as:

$$\lambda(b) = 1 - \textbf{Uncertainty}(b) = 1 - \frac{\overline{V}(b) - \underline{V}(b)}{\overline{V}(b) + \epsilon}, \tag{3.19}$$

and the hybrid base policy used in the rollout simulation becomes:

$$\pi_{hybrid}(b) = \arg\max_a \big(\lambda(b) \cdot \mathbb{E}[\underline{V}(b)] + (1 - \lambda(b)) \cdot \mathbb{E}[\Delta(b)]\big). \tag{3.20}$$

The default upper and lower bounds used in our simulation is similar to the ones in HSVI [50], there are tighter bounds in HSVI2 [51], SARSOP [24] and [60] and this base policy still applies. It could converge much faster than a simple greedy/random base policy at the cost of computation, owing to the powerful superlinear convergence property from Newton's step.

3.3.2 Belief Sampling Strategy

In this section, we propose an efficient sampling strategy for child belief nodes using the hybrid base policy. In HSVI [50], the child node of the current belief point to visit is determined using the IE-MAX heuristic. In particular, it focuses on reachable regions with large gaps, namely regions where approximations are most uncertain, and could have greater potential to improve.

SARSOP [24] proposes an alternative, more efficient strategy that focuses on regions near the subset of belief points that can be reached from an initial belief in the optimal action sequence. The backup operation (refer to Section 3.2.1) is used to select the optimal action to improve the lower bound, and SARSOP included an extra step of propagating the improvements in the lower bound at the current belief back to all previous nodes up to the root node. In particular, a predicted value $\hat{V}(b)$ on the optimal value $V^*(b)$ at a node b will propagate back to the root node. If $\hat{V}$ can improve the lower bound of the root (that is, $\hat{V}$ is greater than the lower bound of the root), b will be expanded and explored down the sampling path. For detailed steps of the sampling strategy and the learning technique used in SARSOP, one can refer to Algorithm 3 in [24]. In summary, the lower bound used in SARSOP is no longer the lower bound based on the alpha vectors computed by the backup operation. As the algorithm iterates and improvements are found in child nodes, an extra term from numerical improvements in child nodes will be added to the nodes along the sampling path. The updated lower bound will be the maximum between the two, and by taking advantage of the propagation of improvements upward to the root, the gap could shrink more quickly to meet the termination criteria, that is, within $\epsilon\beta^{-t}$.

It can be perceived that by using the SARSOP sampling strategy, solving a POMDP with a given initial belief would be much faster, as the improvements in lower bounds of child nodes are backpropagated to the root node. However, the alpha vectors computed through SARSOP could degrade their performance in approximation for regions outside the reachable region with a given initial belief and an optimal policy (i.e. $\mathscr{R}^*(b_0)$). In comparison, the IE-MAX-based sampling strategy used in HSVI and HSVI2 could potentially generate alpha vectors that generalize and scale better in regions outside $\mathscr{R}^*(b_0)$.

The sampling strategy we employ seeks to achieve a balance between the two approaches and is heavily inspired by the Monte Carlo tree search [15]. In particular, we use

the hybrid base policy to select the action at a belief node, and we create a tree-branch-based forward sampling heuristic to select the observation and corresponding successor belief node. This is done to avoid the linear/single sampling path generated with IE-MAX heuristic used in both HSVI and HSVI2, which could encourage the algorithm to explore regions with potential to improve and update the corresponding alpha vectors. Also, we implement the lower bound settings in SARSOP; the actual lower bound at a belief node is determined by the higher one between the value determined by the current set of alpha vectors and the one calculated via backpropagating the improvements in child nodes, which could potentially enhance the rollout simulation with the hybrid policy.

The sampling strategy is described as follows: Given an initial belief node b_0, the action is selected via one-step lookahead maximization with rollout simulation:

$$\tilde{\pi}(b_0) = \arg\max_{a \in A} \big[r(b_0, a) + \beta \sum_{o \in O} \mathbb{P}(o \mid b, a) \tilde{V}_{rollout}(b'(o,a), d) \big]. \tag{3.21}$$

The hybrid base policy used in the rollout simulation to estimate $\tilde{V}_{rollout}$ is as follows:

$$\pi_{hybrid}(b') = \arg\max_{a} \big(\lambda(b') \cdot \mathbb{E}[\underline{V}(b')] + (1 - \lambda(b')) \cdot \mathbb{E}[\Delta(b')] \big), \tag{3.22}$$

with the dynamic balancing factor:

$$\lambda(b') = 1 - \frac{\overline{V}(b') - \underline{V}(b')}{\overline{V}(b') + \epsilon}. \tag{3.23}$$

And $\tilde{a} = \tilde{\pi}(b_0)$ will the action to pick and we update the corresponding alpha vectors using the backup operation in Equation 3.7. In order to decide the next belief node to update, we define two Upper Confident Bound (UCB) [14] terms to construct a tree-branch-based search structure, which are:

$$\textbf{UCB}(o) = \mathbb{P}(o \mid b, \tilde{a}) \cdot \left[\Delta(b'(o, \tilde{a})) + c \cdot \sqrt{\frac{\log N(b)}{1 + N(b,o)}} \right], \tag{3.24}$$

and

$$\textbf{Branch-UCB}(b) = \max_{o} \textbf{UCB}(o) + c_{branch} \cdot \sqrt{\frac{\log N_{root}}{1 + N(b)}}, \tag{3.25}$$

where

- **UCB**(o) is the UCB score for observation o given a belief node b and the corresponding action $\tilde{a}$ selected via the hybrid base policy.
- $N(b)$ is the total visit count for the current belief node b.
- $N(b,o)$ is the total visit count for observation o at b.
- c is the exploration constant to balance the excessive gap size and exploration.
- **Branch-UCB**(b) is a branch-wide UCB score to decide whether to revisit an existing branch from b or explore deeper.
- $\max_o$ **UCB**(o) can capture the potential of a branch based on its child nodes from observations.
- N_{root} is the total visit count at the root b_0.
- c_{branch} is the exploration constant for branch selection, which balances exploration (revisiting previous branches) and exploitation (deep sampling down a promising path).

To sum up, utilizing a tree-branch exploration could avoid overcommitting to deep sampling paths in HSVI and HSVI2, which can balance exploration of less-visited regions and exploitation of promising regions. SARSOP could converge faster to solve problems with a given initial belief, at the cost of degrading the quality of bounds and alpha vectors in unexplored and underexplored regions. Our strategy could be considered as a broader search with optimal allocation of computational resources; by revisiting earlier nodes/branches to ensure bounds and alpha-vectors are refined more globally, which could reduce the risk of missing high-potential regions, therefore improving the overall policy and fit well for an "on-line play" framework.

3.3.3 Step-by-Step Workflow

In this section, we provide a detailed workflow of our algorithm, incorporating the hybrid base policy and the branch-based sampling strategy outlined in the preceding sections.

The search tree consists of nodes representing belief points b with corresponding visit counts $N(b)$, bounds $\overline{V}(b), \underline{V}(b)$, and child nodes linked by observation o. The edges represent action a and observations o with transition probabilities $\mathbb{P}(o \mid b, a)$ and gaps $\Delta(b'(o,a))$. Note that for each node there is only one edge for action (computed by hybrid base policy), and could have multiple child nodes from different observations. This is to avoid spending too much computational resources on one-step lookahead maximization with rollout simulation; successive beliefs are only sampled using the UCB score on observations and branch nodes. In addition, depth-limited exploration is used to stop deep exploration past a certain depth D_{max} and force backtracking on previous promising branches. A detailed workflow of the sampling strategy in our algorithm is summarized in Algorithm 1.

Algorithm 1 Tree Branch Based Sampling Strategy

Initialize : $b_0, \overline{V}(b_0), \underline{V}(b_0)$ ▷ Root node and bounds
Set : D_{max} ▷ Maximum exploration depth
Repeat until termination
For current belief b, **compute**
$\tilde{a} = \tilde{\pi}(b)$ ▷ One-step lookahead with rollout simulation
Perform backup operation with b **and** $\tilde{a}$
Retain only the edge corresponding to $\tilde{a}$
For each observation o, **following** $\tilde{a}$, **compute**
$\mathbf{UCB}(o) = \mathbb{P}(o \mid b, \tilde{a}) \cdot \left[\Delta(b'(o,\tilde{a})) + c \cdot \sqrt{\frac{\log N(b)}{1+N(b,o)}} \right]$ ▷ UCB score of observation
Select $\tilde{o}$ **with highest score**
if $\tilde{o}$ leads to unexplored b' **then**
Initialize b' **with** $\overline{V}(b')$, $\underline{V}(b')$, $N(b')$
Increase visit count for $N(b,\tilde{o})$ **and** $N(b')$
else if Revisiting b' **then**
Perform backup operation to update bounds and alpha-vector
end if
Backpropagate improvements in b' **to parent bounds** $\overline{V}(b)$, $\underline{V}(b)$ **and up to the root**
if Search depth $d > D_{max}$ **then**
Force backtracking and choose branch b **with**
$\mathbf{Branch\text{-}UCB}(b) = \max_o \mathbf{UCB}(o) + c_{branch} \cdot \sqrt{\frac{\log N_{root}}{1+N(b)}}$ ▷ UCB score of branch
else Continue sampling down the path
end if
Terminate when maximum iteration reached/gap at root below a threshold ϵ

3.4 Extension to Coordinator's POMDP

Referring to the Coordinator's POMDP [60] model introduced in Section 3.1, one can perceive that a significant challenge arises because the space $\bar{A}$ of prescriptions is a high-dimensional Cartesian product. For a CPOMDP with many agents, search through such an extraordinarily large action space would be impossible for both the backup operation and the lookahead optimization, preventing the implementation of an "on-line play" policy.

3.4.1 Agent-by-Agent Approach

To address this issue, we adopt an approximate search method known as the agent-by-agent approach, introduced in [8]. The core concept of this approach is to reformulate a typical multi-agent problem into a modified but equivalent problem, where actions are selected sequentially on a *one-agent-at-a-time* basis. In particular, we break down the high-dimensional action $\tilde{A}$ into the sequence of m actions $\gamma^1, \gamma^2, \cdots, \gamma^m$, and between the state transition from $b(\bar{s})$ to $b(\bar{s}')$. Therefore, some artificial intermediate "states" will be created as $(b(\bar{s}), \gamma^1), (b(\bar{s}), \gamma^1, \gamma^2), \cdots, (b(\bar{s}), \gamma^1, \cdots, \gamma^{m-1})$ and the corresponding state transitions. The selection of the last action component γ^m in the intermediate "state" $(b(\bar{s}), \gamma^1, \gamma^2, \cdots, \gamma^{m-1})$ indicates the transition to the subsequent state $b(\bar{s}')$, while receiving the joint reward. This approach ensures that the search space for action selection in the rollout simulation grows linearly with the number of agents, substantially reducing computational complexity compared to the *all-at-once* method, where the search space increases exponentially with the number of agents. The agent-by-agent approach is described as follows:

(3.26)

$$\tilde{\pi}^1(b) = \arg\max_{\lambda^1} \left[r(b, \lambda^1, \pi^2(b), \cdots, \pi^m(b)) + \beta \sum_{o \in O} \mathbb{P}(o \mid b, \gamma) \tilde{V}(b'(o, \lambda^1)) \right],$$

$$\tilde{\pi}^2(b) = \arg\max_{\lambda^2} \left[r(b, \tilde{\pi}^1(b), \lambda^2, \cdots, \pi^m(b)) + \beta \sum_{o \in O} \mathbb{P}(o \mid b, \gamma) \tilde{V}(b'(o, \lambda^1, \lambda^2)) \right],$$

$$\vdots$$

$$\tilde{\pi}^m(b) = \arg\max_{\lambda^m} \left[r(b, \tilde{\pi}^1(b), \tilde{\pi}^2(b), \cdots, \tilde{\pi}^{m-1}(b), \lambda^m) + \beta \sum_{o \in O} \mathbb{P}(o \mid b, \gamma) \tilde{V}(b'(o, \lambda^1, \lambda^2, \cdots, \lambda^m)) \right],$$

where π^i is the base policy for agent i; $\tilde{\pi}^i$ is the rollout policy of agent i that is computed via one-step lookahead with rollout simulation. It can be perceived that the action solved by the rollout policy of agent $i-1$ will be passed on to agent i and used to solve the subsequent rollout policy, leading to the one-at-a-time/agent-by-agent approach. One perspective to interpret this approach is as follows: The first agent executes the one-step lookahead maximization while assuming all other agents following their base policy; the action solved is transmitted to the second agent and use as the decision of the first agent in the first step of action 2's one-step lookahead maximization. Repeat the process for each agent sequentially until the final agent, at which point the system transitions to the next state.

This approach can be used in the off-line session to train the base policy. However, when agents are deployed and communications among agents are prohibited, the autonomous multi-agent rollout approach [8] can be implemented for real-time planning in a decentralized multi-agent setting. In particular, each agent can store precomputed signaling policies from other agents trained off-line with agent coordination. In a decentralized environment where agents are allowed to communicate but have access to the state signal, these precomputed approximations can serve as action components for the preceding agents (in the agent-by-agent approach), allowing each agent to compute its own actions in parallel and asynchronously.

More precisely, referring to Equation 3.26, the signaling policies of $\hat{\pi}^1$ to $\hat{\pi}^i$ can be used to solve the rollout policy $\tilde{\pi}^{i+1}$, instead of communicating $\tilde{\pi}^1,\cdots,\tilde{\pi}^i$ directly to the agent $i+1$. It is given as:

$$\begin{aligned}\tilde{\pi}^{i+1}(b) = \arg\max_{\lambda^{i+1}} \Big[& r(b,\hat{\pi}^1(b),\hat{\pi}^2(b),\cdots,\hat{\pi}^i(b),\lambda^{i+1},\pi^{i+1}(b),\cdots,\pi^m(b)) + \\ & \beta \sum_{o\in O} \mathbb{P}(o \mid b,\gamma)\tilde{V}(b'(o,\hat{\pi}^1(b),\cdots,\lambda^{i+1}))\Big].\end{aligned} \tag{3.27}$$

This approach enables all actions to be computed asynchronously and in parallel without direct communication, as long as the signaling policy values $\hat{\pi}^1(b),\cdots,\hat{\pi}^{m-1}(b)$ are precomputed and accessible to all agents. As noted in [8], the signaling policy can be simply the base policy, but this carries the risk of not achieving the policy improvement property. The author proposed selecting signaling policies that approximate the multi-agent rollout policy by utilizing a neural network trained on a dataset generated

through simulating rollouts of base policies. From the coordinator's perspective, the one-at-a-time approach can still be done in sequence without communication among all agents. However, there will be delays in computing "prescriptions" for each agent, which could potentially degrade the performance in real-time.

3.4.2 Updated Bounds in Coordinator POMDP

Since the Coordinator's POMDP has a high-dimensional action space, it can be considered as a multi-agent POMDP. We included a proof on the validity of the approximation with alpha vectors in multi-agent POMDP based on the original proof provided in [52]. In particular, the value function of a multi-agent POMDP is still piecewise linear and convex, and see Appendix A3 for details.

By incorporating the agent-by-agent approach to decompose the search space, additional lower bounds based on alpha vectors are required to represent the approximations in intermediate artificial "states". The authors of [60] also introduced a Coordinator Heuristic Search Value Iteration (CHSVI) with lower bound functions corresponding to each intermediate artificial "state", which is similar to the sequential action selection paradigm of the agent-by-agent approach, which is as follows:

Considering a CPOMDP with 2 agents, each time step t is extended into three stages: $(t,0),(t,1),(t,2)$, with the corresponding state space $\bar{S},\bar{S}\times\bar{A}^1,\bar{S}\times\bar{A}^1\times\bar{A}^2$. In [60], the prescription is defined as $\gamma^i(a^i \mid m^i) := \mathbf{1}_{\gamma^i(m^i)=a^i}$ for $\gamma^i \in \bar{A}^i$ (deterministic mapping), and the belief update function for the first two stages are $[\tau^o(b,\gamma^1)](s,\bar{a}^1) = b(s)\gamma^1(a^1 \mid m_s^1)$, and $[\tau^1(b,\gamma^2)](s,\bar{a}^1,\bar{a}^2) = b(s,a^1)\gamma^2(a^2 \mid m_s^2)$, respectively. The belief update in the last stage $[\tau^2(b,o)](s')$ is calculated via the belief update formula in Appendix A2. Based on the different stages within a time step t, three sets of alpha vectors are used as lower bound functions for each stage in CHSVI. Let L^l represent the lower bound function and V^l denote the corresponding alpha vectors for stage l. Using T^l as the Bellman operator for stage l, the lower bound update in CHSVI is given as:

For stage $l = 0,1$, the coordinator selects a prescription for agent $i = l+1$. The Bellman update of the lower bound at $b \in \Delta\bar{S}^l$ is:

$$
\begin{aligned}
T^l L^{l+1}(b) &= \max_{\gamma^i \in \bar{A}^i} L^{l+1}(\tau^l(b, \gamma^i)) \\
&= \max_{\gamma^i \in \bar{A}^i} \max_{\alpha \in V^i} \sum_{\bar{s}^l, a^i \in \bar{S}^l \times A^i} b(\bar{s}^l) \gamma(a^i \mid m^i_{\bar{s}^l}) \alpha(\bar{s}^l, a^i),
\end{aligned} \tag{3.28}
$$

and

$$
\begin{aligned}
\gamma^{i,\alpha}(m^i) &:= \arg\max_{a^i \in A^i} \sum_{\bar{s} \in \bar{S}: m^i_{\bar{s}^l} = m^i} b(\bar{s}^l) \alpha(\bar{s}^l, a^i) \quad \forall m^i \in M^i, \alpha \in V^i \\
J(\alpha) &:= \sum_{\bar{s} \in \bar{S}^l} b(\bar{s}^l) \alpha\big(\bar{s}^l, \gamma^{i,\alpha}(m^i_{\bar{s}^l})\big) \quad \alpha \in V^i \\
\alpha^* &:= \arg\max_{\alpha \in V^i} J(\alpha) \\
\gamma^{i,*} &:= \gamma^{i,*\alpha^*},
\end{aligned} \tag{3.29}
$$

where for each fixed $\alpha \in V^i$, the optimization over $\gamma^i \in \bar{A}^i$ can be decomposed into $|M^i|$ optimizations problems over A^i. Consequently, the following new $\alpha^b(\bar{s}^l)$ is added to V^l,

$$
\alpha^b(\bar{s}^l) = \sum_{a^i \in A^i} \gamma^{i,*}(a^i \mid m^i_{\bar{s}^l}) \alpha^*(\bar{s}^l, a^i) \quad \forall s^l \in S^l. \tag{3.30}
$$

Calculations in these two stages are greedy choices for immediate reward in intermediate stages $(b, \bar{A}^1)$,$(b, \bar{A}^1, \bar{A}^2)$. The last stage considers the probability of transition and the state of the system evolves to b', just as the agent-by-agent approach [8], and is updated by means of the backup operation with actions chosen in the first two stages.

Remark 3.2: The lower bound updates in our algorithm differ slightly from CHSVI, as we included a one-step lookahead with rollout simulation at the stage of the last agent. For the first agent in sequence, we utilize the same update strategy as CHSVI, as solving the optimization problem of a prescription-based action space is computationally expensive. In other words, we let the first agent follow a greedy base policy on the immediate reward, and only conduct rollout simulation with shallow depth at the end of stage.

3.5 Experiment

In this section, we evaluate our algorithm using two benchmark problems, *DecTiger* and *MultiCast*, and compare the simulation results with those reported in CHSVI [60].

The Decentralized Tiger (DecTiger) benchmark [34] is a widely studied Dec-POMDP problem involving two agents positioned in front of two (or more) doors. Behind one door lies a treasure, while the other conceals a tiger. The state is defined by the location of the tiger, behind the left door (s_l) or the right door (s_r), each occurring with equal probability. Agents can perform one of three actions: open one of the two doors, or listen for the tiger's roar, which provides probabilistic information about its location. The reward structure penalizes opening the door with the tiger, while opening the treasure door yields a positive reward. If both agents open the treasure door simultaneously, the reward is even higher. However, any door-opening action resets the tiger's position randomly, based on a uniform probability distribution.

MultiCast [60] is a two-agent broadcast system inspired by the models in [29] and [41]. Each agent manages a buffer with limited capacity, initially empty, where packets arrive independently with a given probability. Packet arrivals are independent of both the history and the other agent's arrivals. If a packet arrives when the buffer is full, it is dropped, resulting in a penalty. The objective of both agents is to transmit packets over a shared broadcast channel. A packet is sent and removed from the buffer successfully if only one agent transmits. However, if both agents transmit simultaneously, a collision occurs, causing both transmission attempts to fail. Each transmission, successful or not, incurs a cost and agents also face a delay penalty for packets that remain in their buffers at each time step. Future penalties and costs are discounted, and each agent has access only to its private packet arrival history, with no information about the other agent's history.

3.5.1 Simulation Environment

We refer to the simulation settings proposed in [60], and use the CHSVI algorithm as our baseline for testing.

DecTiger: A parameterized extension of the DecTiger model is considered with three

a1\a2	listen	open tiger door	open other door
listen	-2	-101	$\frac{20}{N}-1$
open tiger door	-101	-50	-100
open other door	$\frac{20}{N}-1$	-100	$\frac{40}{N}$

Table 3.1: Reward function of DecTiger.

parameters (N, d, β). In particular, two agents face N closed doors with a delay of d steps to share private information and a discount factor β. Behind one of the doors is a tiger, whereas the remaining doors conceal treasures. At each time step, the agent can choose from $N+1$ actions: opening one of the doors or listening for the tiger's roar. However, a cost is added to listening and is flawed: with a probability of $\frac{0.85}{0.7+0.15N}$ to hear the roar from the correct door and from any other door with a probability of $\frac{0.15}{0.7+0.15N}$. The immediate rewards of the agents are summarized in Table 3.1.

Initially, the tiger is placed behind a door following a uniform random distribution. The state remains unchanged if both agents choose to listen simultaneously. In this case, at the next time step, each agent receives conditionally independent observations based on a predefined distribution. However, if either agent opens a door, the tiger's location is reset to a new uniform random position. From that point onward, all subsequent observations become independently uniform and unrelated to the tiger's actual location. Unlike the original DecTiger setup [34], where the actions and observations of each agent are entirely private, the CPOMDP model incorporates a delayed information sharing mechanism. In this model, agents can share their private information with each other after a delay d. In particular, at time t, each agent can access the actions and observations of the other agent from any time before (but does not include) $t-d$, while any action taken at or after $t-d$ along with corresponding observations remain private at time t.

MultiCast: The two-agent multi-access system model is based on previous works in [60] [29] [41]. It is parameterized by $(C^1, C^2, p_1, p_2, \beta)$ where C^i indicates the buffer size of agent i and β is the discount factor. A packet will arrive in agent i's buffer with probability p^i at each time step, independently of the history and packet arrival events of the other agent. Both agents will attempt to transmit, and if only one agent transmits, the packet will be delivered successfully and removed from the buffer. However, a collision will occur when both agents transmit at the same time, causing both to fail. There

	States	Observations	Prescriptions(approximated)
DecTiger(2,1,β)	74	37	3^{14}
DecTiger(3,1,β)	435	145	4^{26}
MultiCast(8,8,0.2,0.4,β)	64	4	2^{18}
MultiCast(16,16,0.2,0.4,β)	256	4	2^{34}

Table 3.2: Joint space on different settings.

will always be a transmission cost $c_T = 0.5$ regardless of whether or not transmission attempts succeed. In addition, agents are penalized by 1 for each time unit in which a packet remains in the buffer.

Since each agent is aware of its own history of packet but not the other agent's, and other agent's actions are observable after taken, this system can be rigorously modeled as a CPOMDP. Studies in [29] and [41] have demonstrated that the number of packets in each agent's buffer serves as sufficient private information for decision making. As a result, the state space is defined as the joint set that represents the number of packets in the buffer of each agent. Each agent has two possible actions: transmit or not transmit. Although agents can observe each other's actions, they cannot access each other's histories. Consequently, the common information corresponds to the joint action space of both agents, consistent with the control-sharing scenario outlined in the common history framework [35]. Additionally, the instantaneous reward function is given as:

$$r(s,a) = \sum_{i=1}^{2} r_i(s_i, a_i). \tag{3.31}$$

One point to note is that although both benchmarks have seemly small state space based on settings, the actual joint space is much larger according to [62], and is shown in Table 3.2.

3.5.2 Simulation Results

We compare our algorithm with CHSVI [62] on the two benchmarks in various settings. The termination condition is either that the target gap at the root node drops below 0.5 or that the execution time exceeds the maximum runtime. For small-scale problems, the maximum runtime is set to 600 seconds and to 1200 seconds for settings with larger

state space.

We first test and compare performance on relatively small-scale problems, with $(2, 1, 0.9)$ and $(2, 1, 0.95)$ in DecTiger, and are shown in Figure 3.1. It can be seen that our algorithm can converge slightly faster than the baseline, the difference in convergence rate is not evident when the discount factor is lower. However, when we increase the discount factor to 0.99, both algorithms were unable to converge within the maximum runtime, and our algorithm still shows a faster convergence rate, which can be seen in Figure 3.2.

We then increase the number of doors N to three and test again with different discount factors. The results can be seen in Figure 3.3. Although both algorithms cannot converge at the root node within the maximum run-time, our algorithm can still achieve a lower gap when compared with the baseline algorithm.

In the *MultiCast* benchmark, our algorithm performs slightly better than the baseline algorithm, as illustrated in Figure 3.4. This is partially due to the fact that DecTiger has a much larger observation space than Multicast, so the benefits from our improved sampling strategy might not be obvious when choosing the belief node.

3.6 Summary

In this chapter, we present our on-line planning algorithm for solving the CPOMDP model in decentralized multi-agent environments from the perspective of "on-line play" policies. The core of the algorithm is a hybrid policy inspired from key components in approaches including point-based value iteration, heuristic search value iteration, one-step lookahead optimization with rollout simulation, and Monte Carlo tree search. The proposed algorithm has been evaluated on two benchmark problems and showed promising results.

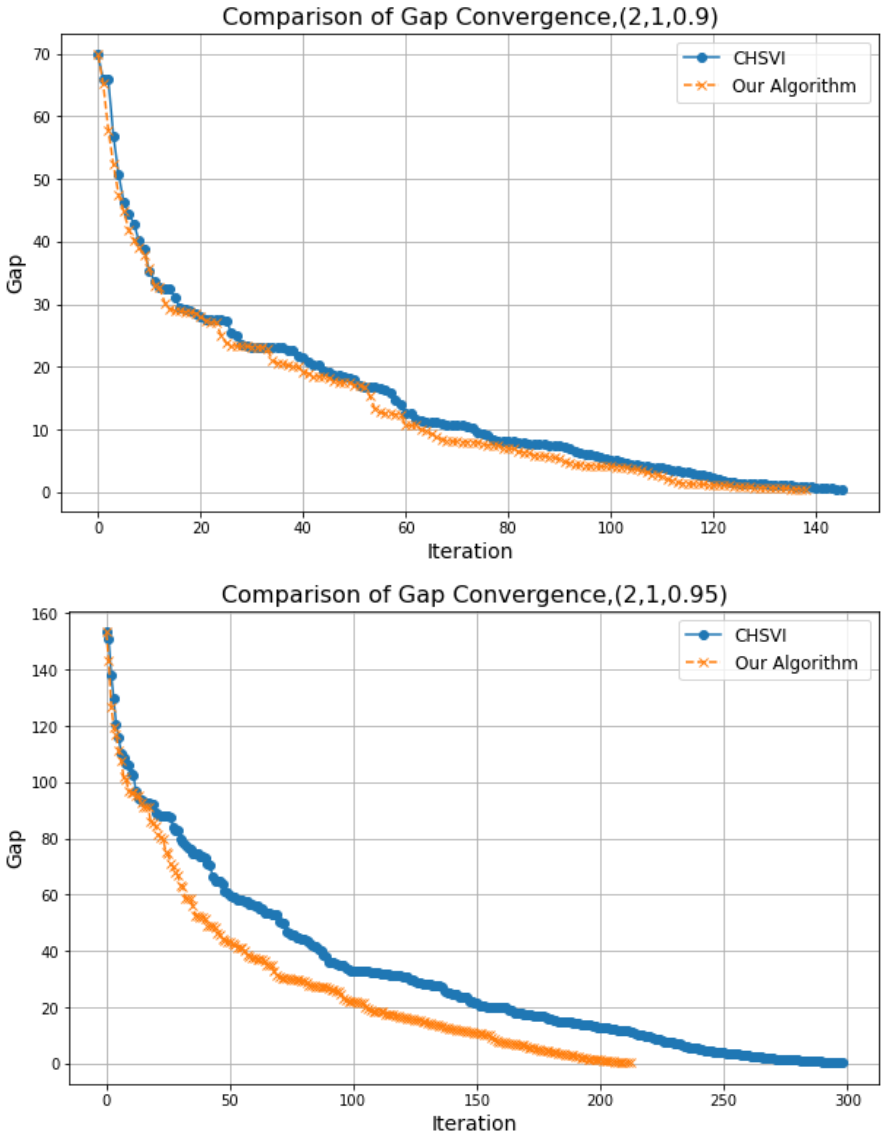

Figure 3.1: Simulation results on DecTiger: (2,1,0.9) and (2,1,0.95)

.

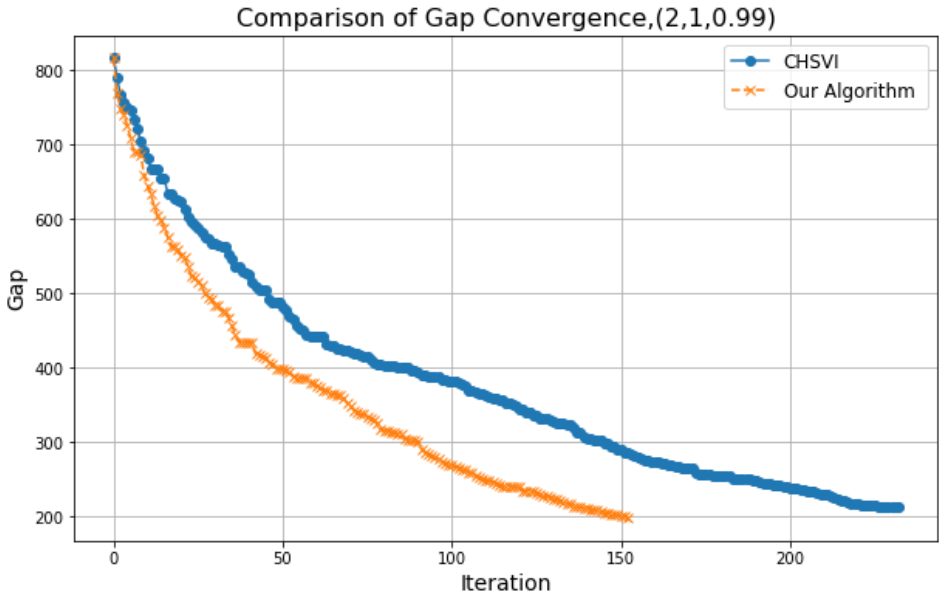

Figure 3.2: DecTiger with (2,1,0.99)

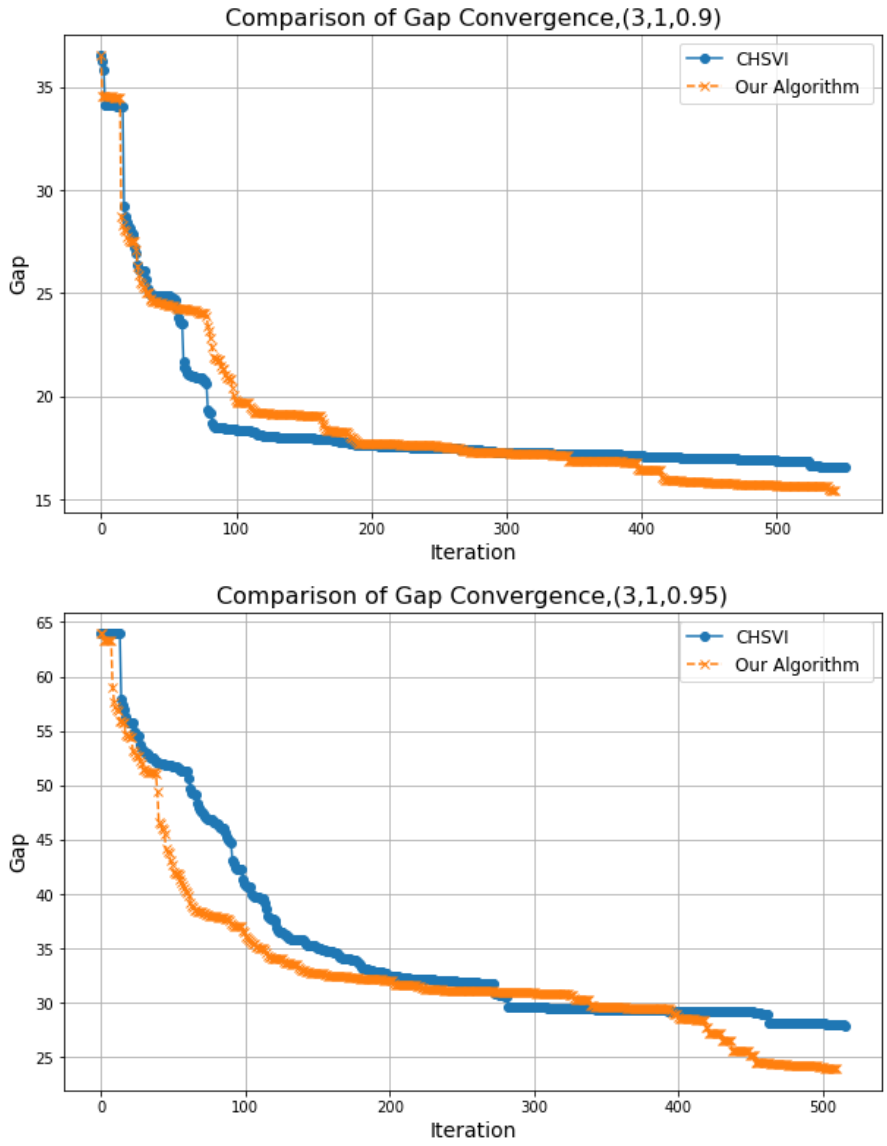

Figure 3.3: Simulation results on DecTiger: (3,1,0.9) and (3,1,0.95)

.

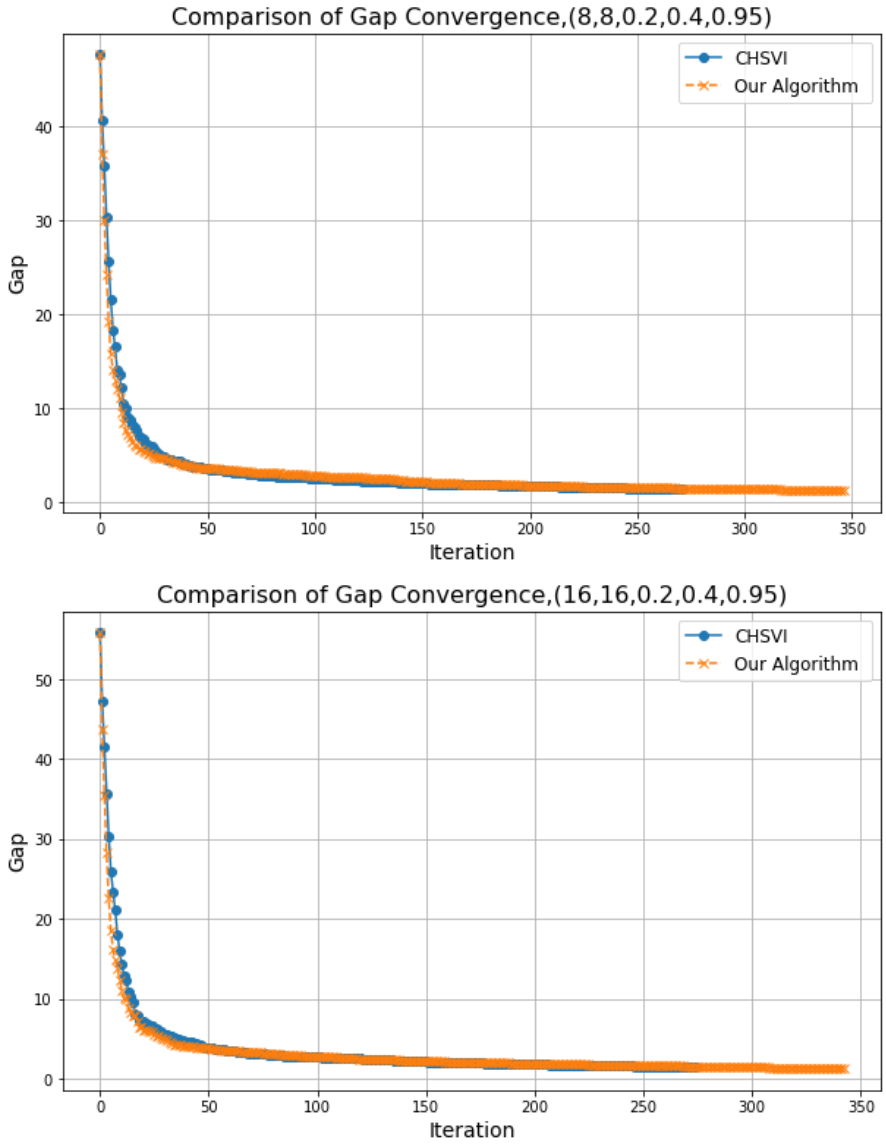

Figure 3.4: Simulation results on MultiCast: (8,8,0.2,0.4,0.95) and (16,16,0.2,0.4,0.95) .

CHAPTER 4

APPLICATION IN MICROGRID

In this section, a case study of our algorithm is conducted on a microgrid model, which deals with distributed energy resources (DERs) within the microgrid to participate in the frequency regulation market. A discussion of the background information on the frequency regulation market is provided, and we have formulated and modeled the corresponding problem and developed a method to incorporate a detailed battery aging model into the POMDP framework. The proposed models are applicable to many real-world storage systems to reduce the costs incurred by battery aging.

4.1 Frequency Regulation Market

The Australian national electricity market works as a spot market; the price is determined by the Australian Energy Market (AEMO) Operator based on power generators to meet demand. All the market participants (generators) offer their bids or a certain amount of power to be supplied to the market for a particular price at a specified period. The AEMO's central dispatch engine ranks generators' offers from the least to the most expensive and determines which generators to dispatch during each 5-minute dispatch interval to instantaneously balance demand and supply. The generators must bid to supply in five minute intervals, and all generators supplied power during the period are paid based on the spot price over that period. Renewable energy usually has a higher priority in dispatch as the cost of generation is lower compared to conventional generators.

The frequency regulation market addresses frequency deviations caused by imbalances between generation and load within electric power systems. To mitigate these deviations, frequency feedback control is provided by Automatic Generation Control (AGC) signals, with the aim of balancing load and generation in real time. Traditionally, this service has been delivered by large energy resources such as fossil fuel-based power plants. As the need to reduce carbon emissions grows, the industry has initiated a trend toward integrating distributed energy resources (DERs) into the system for frequency regulation services. The fast response and high ramp capability of DERs are particularly effective for frequency regulation, but they often face limitations in capacity and may not meet the minimum requirements for participation directly in the market. A promising solution is to aggregate groups of DERs through aggregators, which can act as virtual power plants (VPPs). These VPPs coordinate with the system operator to provide frequency regulation services collectively, overcoming individual DERs' capacity constraints.

From the MARL perspective, the aggregator of DERs can be considered as the agent that is required to balance the frequency deviation within a microgrid. The uncertainty within the environment is caused by factors such as renewable resources, electricity prices, power demand, and so on, which are mostly stochastic but could be model by a probability distribution model. In addition, reward functions can be defined by cost models related to supply mismatches and device aging that are determined by DER operations.

4.2 Current Practice

According to [53], there are three key stages in the current practice of frequency regulation by individual resources: market clearance, disaggregation of the regulation signal, and real-time tracking of the regulation signal. The market is managed by a Regional Transmission Organization (RTO) that coordinates the output of the participants centrally. It is responsible to calculate and assign regulation signals (actions) to individual resources in order to maintain the system frequency at a nominal value, thereby restoring power balance to the grid.

The current approach encounters significant challenges when aggregating variable

and intermittent distributed energy resources (DERs) such as renewable generation, electric vehicles (EVs), and thermostatically controlled loads (TCLs). This is primarily because the allocation of regulation signals assumes a fixed number of resources with static generation capacities. As noted in [53], several critical issues must be addressed: 1) Within microgrids, the RTO must identify aggregate cost functions and regulation capacity limits, accounting for the costs and flexibilities of DERs; 2) Capacity bids must incorporate the uncertainties inherent in the variable and intermittent nature of DERs within microgrids; and 3) Mileage bids must reflect the ramp rate dependencies of individual DERs, as their ramp rates vary and are not constant like those of conventional energy resources used in current practices.

4.3 Problem Formulation

Based on the current practice in the frequency regulation market, we consider a microgrid comprising a group of aggregators, each associated with controllable and uncontrollable loads. These include battery energy storage systems, EV charging stations, demand response units with deferrable appliances, and user load demands within the microgrid. Each aggregator functions as an agent, making the microgrid a multi-agent system.

We aim to address the following two problems in the current practice under this microgrid environment: bids of individual aggregators in the market clearance and the allocation of the regulation signal of each aggregator, which can be formalized as:

- Each aggregator must specify the maximum up/down regulation capacity it can provide, the unit cost for delivering this regulation, and its ramp rate, which defines how quickly it can adjust its power level within a regulation period.
- The coordination problem involves determining the set points for each aggregator at every instant during the regulation period.

4.3.1 Microgrid Model

Consider a microgrid represented by a directed graph $G = (V, E, A)$, where V is the set of vertices; $E \subseteq \{i, j\}$, $i, j \in V$ and $i \neq j$, is the set of edges. A is an incidence matrix

that maps every edge to an ordered pair of vertices. The microgrid consists of n nodes described by V, where each node corresponds to an aggregator within the microgrid. The first node (bus) is connected to the utility grid via a tie line. The remaining nodes are partitioned as $V_{cl} \cup V_{ul}$ where the set of controllable nodes is expressed by V_{cl} and the set of uncontrollable nodes is V_{ul}.

The power level at each controllable node is represented as P_{cl}, with V_{ul}^0 denoting the baseline generation / consumption. P_{ul} represents the power level on the nodes that are uncontrollable, and the power of the microgrid cable is indicated by P_{tl} with the baseline value P_{ul}^0. Over the period of providing frequency regulation service, the value of the tie line power is:

$$P_{tl} = P_{ul}^0 + P_{fr}, \tag{4.1}$$

where P_{fr} is the allocated AGC signal for frequency regulation provided by the RTO. Assume there is no overlapping loops in power flow, the corresponding equations inside the microgrid will be:

$$[P_{tl} \quad P_{cl}^T \quad P_{ul}^T]^T = M\omega, \tag{4.2}$$

$$|\omega| \leq \bar{\omega}, \tag{4.3}$$

where P_{cl} and P_{tl} represent the vectors of controllable and uncontrollable nodes, respectively. M is the incidence matrix of the microgrid graph, ω denotes the vector of line power flows, and $\bar{\omega}$ indicates the vector of maximum allowable power flows.

The microgrid must solve an optimization problem to determine the maximum upward or downward regulation it can offer, which constitutes the capacity bid. For up regulation, the microgrid's power consumption must be less than the baseline power. Also, determining the capacity bid involves minimizing P_{tl} while satisfying the power flow constraints. Assuming the uncontrollable load remains constant throughout a single step of the regulation event, the optimization problem is formulated as:

$$
\begin{aligned}
\underset{P_{cl},\omega}{\text{minimize,}} \quad & P_{tl} \\
\text{subject to} \quad & [P_{tl} \quad P_{cl}^T \quad P_{ul}^T]^T = M\omega \\
& \underline{P_{cl}} \leq P_{cl} \leq \overline{P_{cl}} \\
& |\omega| \leq \bar{\omega}.
\end{aligned} \tag{4.4}
$$

$\underline{P_{cl}}$ and $\overline{P_{cl}}$ represent the minimum and maximum allowable power for the controllable loads, respectively. If P_{cl}^* is the optimal solution to the minimization problem, the maximum up regulation, or the capacity bid is $P_{fr} = P_{cl}^* - P_{tl}^0$. The maximum downward regulation can be determined by solving a similar optimization problem, modified to account for scenarios where the microgrid's power consumption exceeds the baseline power.

4.3.2 POMDP Formulation

The optimization problem deals with one regulation event only. We introduce the following dynamic system model to obtain the bid capacity over a regulation period with N stages.

4.3.2.1 States of the System

In this section, we focus on the uncontrollable loads, which are the net amount of renewable power and the power demand at all the uncontrollable load nodes P_{ul}. Let $S_i^{RN}(k)$ be the renewable power available at time k in the uncontrollable node i, and $S_i^{PD}(k)$ be the load demand power at time k on uncontrollable node j, with $i, j \in P_{ul}$.

The variable W_k is the vector that contains exogenous stochastic processes, which are the change in the available net renewable power and load demand, including the change in the net renewable power at the uncontrollable load node i between $k - \Delta k$ and k, and the change in the demand for load power at the uncontrollable load node j between $k - \Delta k$ and k.

It is defined as the information available between $k - \Delta k$ and k. So, the system transition model for uncontrollable nodes will be:

$$S^{ul}(k+1) = S^{ul}(k) + W_{k+1}. \tag{4.5}$$

W_k captures the stochastic information over the regulation period and only changes the states S^{ul} of all uncontrollable nodes.

4.3.2.2 Stochastic POMDP

The transition model of controllable nodes is as follows:

$$S^{cl}(k+1) = S^{cl}(k) + \phi^T a_k, \tag{4.6}$$

where ϕ^T is an incidence column vector that models the dynamics of the controllable load nodes. We use f_k to represent the state transition function and $S^{cl}(k+1) = f_k(S^{cl}(k), a_k, \phi)$, which can be the state transition probability.

Recall that P_{tl} is the tie line power to the utility grid that must be minimized to obtain the maximum capacity bids. On one regulation step, P_{tl} can be solved by a directed graph model of the microgrid.

P_{tl} and P_{cl} are the power injections at all controllable and uncontrollable nodes, which are the states of the system. Let $g_k(S^{cl}(k), S^{ul}(k), a_k, \phi, W_{k+1})$ be the stage cost for the following minimization problem, and we formulate the following minimization problem:

$$\begin{aligned} \underset{P_{cl},\omega}{\text{minimize}} \quad & \sum_{k=1}^{N-1} P_{tl}(k), \\ \text{subject to} \quad & [P_{tl} \quad P_{cl}^T \quad P_{ul}^T]^T = M\omega \\ & \underline{P_{cl}} \le P_{cl} \le \overline{P_{cl}} \\ & |\omega| \le \bar{\omega}. \end{aligned} \tag{4.7}$$

Assume a regulation period with N stages, and a policy $\pi = \mu_1, \mu_2, \cdots, \mu_N$ is a list of functions that map the state $S^{cl}(k)$ into control $a_k = \mu_k(S_k)$. The cost of policy π at the first state S_1 of the whole microgrid system is expressed as:

$$J_\pi(S_1) = \mathbf{E}\{g_N(S_N) + \sum_{k=1}^{N-1} g_k(S^{cl}(k), S^{ul}(k), a_k, \phi, W_{k+1})\}, \tag{4.8}$$

and the Bellman equation to solve the optimal policy $\pi^* = \mu_1^*, \mu_2^*, \cdots, \mu_N^*$ is:

$$\mu_k^*(S_k) = \arg\min \mathbf{E}\{g_k(S^{cl}(k), S^{ul}(k), a_k, \phi, W_{k+1}) + J_{k+1}^*(S_{k+1})\}. \tag{4.9}$$

Calculating the optimal policy is equivalent to solving the minimization problem 4.7 where we obtain the minimum tie line power to the utility grid over the regulation period, and thus the maximum capacity bids can be obtained.

This optimization problem can naturally be represented as a multi-agent POMDP: the uncertainties in state transitions are determined by the parameters W_{k+1} and ϕ, and the reward function is related to the minimization problem 4.7. This formulation assumes a centralized control strategy and we can convert it into a decentralized CPOMDP by assuming a delay in the sharing of state and action information in each aggregator. Nevertheless, the battery aging cost model is not considered in this model, and we introduce a method to incorporate such a model in the next section.

4.3.3 Cost of Battery Aging Model

A significant challenge in microgrid operation lies in optimizing battery performance by minimizing its loss in lifetime consumption. Incorporating a cost associated with battery lifetime consumption into the reward function of a POMDP is highly desirable. However, estimating battery lifetime consumption depends on rain-flow algorithms, which are based on charge/discharge cycles and can be difficult to integrate into the additive reward structure of a POMDP. In this section, we present a model to address this issue, partially based on our previous work [67].

Our battery cost model is based on the battery lifetime framework described in [21]. A notable feature of this model is its ability to identify "half-cycles," which are defined as the differences between two consecutive "local extrema" on the battery energy level curve. These half-cycles provide critical insights into the battery's charging and discharging

patterns, allowing for an accurate assessment of its degradation over time.

4.3.3.1 Model of Counting Battery Charge and Discharge Cycle

The battery cost model is based on the cycle counting method introduced in [21], and is given as follows:

$$T_{cycle} = \frac{N_d^{fail}}{w \cdot n_d^{day}}, \tag{4.10}$$

where N_d^{fail} represents the maximum number of charge-discharge cycles, or "full cycles" that the battery can perform at a given depth of discharge (DOD). n_d^{day} denotes the number of daily operation cycles the battery undergoes at a specific DOD. Using w, the average number of operating days within a year, the battery life can be estimated by solving T_{cycle}, expressed in years. Moreover, N_d^{fail} is a function of the DOD, provided by battery manufacturers, and is defined as:

$$N_d^{fail} = f(d) = N_{100}^{fail} d^{-kp}, \tag{4.11}$$

where d is a value of DOD between 0 and 100%, N_{100}^{fail} is the maximum number of cycles that a battery can experience at 100% DOD, and kp is a constant determined by the specifications of battery.

Assuming that the battery undergoes n_d cycles in a DOD d, its cycle life loss Ls_{cycle} can be expressed as:

$$Ls_{cycle} = \frac{n_d}{f(d)} \times 100\%. \tag{4.12}$$

Consequently, for the same rate of cycle life loss Ls_{cycle}, the equivalent number of 100%-DOD cycles, denoted as n_{100}^{eq}, corresponding to n_d cycles at d DOD, can be calculated using the following equation:

$$\frac{n_{100}^{eq}}{N_{100}^{fail}} = Ls_{cycle} = \frac{n_d}{f(d)}. \tag{4.13}$$

Substituting the function for the maximum number of charge-discharge cycles into the equivalent cycle number equation, we get:

$$n_{100}^{eq} = n_d \cdot d^{kp} \ . \tag{4.14}$$

Using this equation, the depth of discharge (DOD) and the number of various cycles can be calculated and converted to an equivalent value. Assuming that the battery operates in a consistent pattern over a given period, its corresponding lifetime can be estimated using Equation (4.10).

4.3.3.2 Counting Half Cycles

Rather than counting full cycles, the authors of [21] define a half-cycle as the interval between two consecutive local maxima and minima in the energy level, representing the switching points between charging and discharging actions. Let E_{max} represent the rated energy capacity of the battery, and E_k be the energy level of the battery at the end of the k-th half-cycle. The corresponding DOD, denoted as d_k^{half}, can be expressed as:

$$d_k^{half} = \left| \frac{E_k - E_{k-1}}{E_{max}} \right|. \tag{4.15}$$

And the equivalent 100%-DOD cycle number for K half-cycles can be derived from Equation (4.15):

$$n_{100}^{eq} = \sum_{k=1}^{K} 0.5 \cdot (d_k^{half})^{kp}. \tag{4.16}$$

4.3.3.3 Cycle Life Cost Model

Since a half-cycle represents a single charge or discharge action between local extrema (the switching points between charging and discharging), the term $Ls_{cycle}(\%)$, representing the cycle life loss percentage, is an appropriate metric to model battery life costs. Considering a replacement cost R_c incurred at the end of the battery's lifespan, and a cycle life loss $Ls_{cycle}(w)$ over a period w, the corresponding cost of battery lifetime consumption during this period is the product of these two terms. In our case, the period W corresponds to the actual duration of each planning horizon, and the cycle life loss

is determined by the battery's actions within this horizon. For a single half-cycle with depth d_k^{half}, the cost of cycle life loss C_{loss} can be expressed as:

$$C_{loss}(d_k^{half}) = \frac{n_{100}^{eq}}{N_{100}^{fail}} R_c = \frac{0.5 \cdot (d_k^{half})^{kp}}{N_{100}^{fail}} R_c \ . \tag{4.17}$$

where N_{100}^{fail} represents the total number of full 100%-DOD cycles the battery can complete over its lifetime. The optimization problem can be formulated to minimize both the cycle life cost and the energy cost while meeting the power demand. This is accomplished by making decisions on battery operations based on varying electricity prices.

4.3.4 Updated Battery Cost Model for Dynamic Programming

In this section, we present an updated model for calculating the cost of loss in battery life, which can be applied in the dynamic programming approach. This model is based on our previous work published in [67].

The standard dynamic programming algorithm operates recursively, adhering to the Bellman principle of optimality [7]. The process begins at the final step of the planning horizon, evaluating the two adjacent states within that step. For each preceding state, the algorithm determines the action that minimizes the cost for the current control step and records the corresponding cost. The algorithm then moves to the second to last step of the planning horizon and repeats this procedure. The previously stored values, representing the costs of the successor states of the prior step, are utilized to calculate the optimal costs for the current step. By the time the algorithm reaches the initial step of the planning horizon, it will have computed the optimal costs for all starting states throughout the horizon.

A key principle of the dynamic programming approach is that the problem's cost must be cumulative/additive. In our case, the energy trading cost is proportional to the amount of power exchanged with the main grid and accumulates over the control steps. However, the cost associated with the battery cycle life loss depends on the switching points between the battery's charge and discharge actions, which may span multiple steps. As a result, the cost of battery life loss, derived from Equation (4.17), cannot always

be calculated directly by iterating over adjacent states in the DP algorithm. To overcome this limitation, we propose a method for computing the battery life loss cost that is compatible with the DP framework. This method accounts for consecutive charging or discharging actions across multiple steps while ensuring the additivity of costs required by the dynamic programming approach.

Refer to Figure 4.1. Let BC and CD represent the actions of the battery in steps 2 and 3, respectively, and let A denote the state-of-charge at step 1, which will be decided later. Based on Equation (4.17), the cost over step 2 and 3, or the "cost-to-go", is given by:

$$\frac{0.5R_c}{N_{100}^{fail}}(|B-C|)^{kp}+\frac{0.5R_c}{N_{100}^{fail}}(|C-D|)^{kp}. \tag{4.18}$$

If A is higher than B, the cost over steps 1, 2 and 3 becomes:

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-C|)^{kp}+\frac{0.5R_c}{N_{100}^{fail}}(|C-D|)^{kp}, \tag{4.19}$$

as AB and BC are two subsequent discharge actions, with C being the local minimum.

On the other hand, if A is lower than B, AB will become charge action, and the cost over the three steps is:

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-B|)^{kp}+\frac{0.5R_c}{N_{100}^{fail}}(|B-C|)^{kp}+\frac{0.5R_c}{N_{100}^{fail}}(|C-D|)^{kp}. \tag{4.20}$$

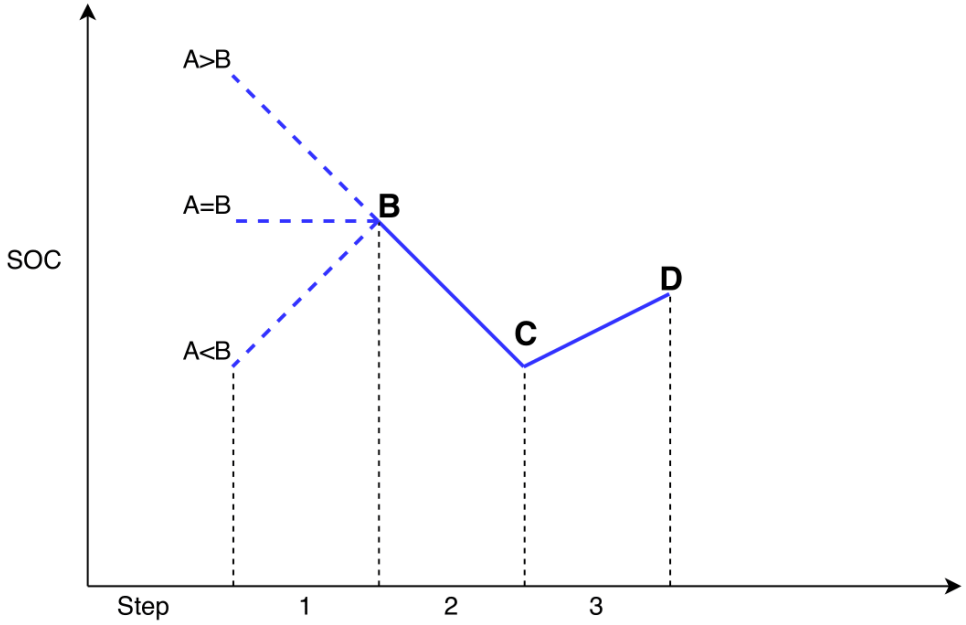

Figure 4.1: State of charge (SOC) profile used for illustration.

Since the dynamic programming algorithm evaluates only the cost within the current step, we propose the following method to calculate the life cycle cost: Consider a step k within the planning horizon of N steps, where $k = 0, 1, \ldots, N-1$. Based on the battery model introduced earlier, $E_B(k)$ and $E_B(k+1)$ represent the state of charge (SOC) of the battery at the beginning and end of step k, respectively. We define σ_k as the "subsequent local extrema" of step k. This refers to the next local extrema (either a maximum or a minimum) related to $E_B(k)$, which is determined by the actions after step k.

Specifically, this σ_k represents the next switching point between the charging and discharging actions as observed in step k, and the cost of loss of battery life cycle in step k can be defined as:

$$C_{loss}(k) = \frac{0.5R_c}{N_{100}^{fail}}\left(\left|\frac{E_B(k)-\sigma_k}{E_{max}}\right|\right)^{kp} - \frac{0.5R_c}{N_{100}^{fail}}\left(\left|\frac{E_B(k+1)-\sigma_k}{E_{max}}\right|\right)^{kp} \tag{4.21}$$

where kp is the parameter that governs the nonlinearity of cycle life degradation, and N_{100}^{fail} is the number of 100%-DOD cycles the battery can sustain before failure.

As illustrated in Figure 4.1, if the states satisfy $A \geq B$, then σ_1 will correspond to the state C. Substituting the values into Equation (4.21), the cost in this step is:

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-C|)^{kp} - \frac{0.5R_c}{N_{100}^{fail}}(|B-C|)^{kp}, \tag{4.22}$$

and the cost over the three steps becomes

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-C|)^{kp} - \frac{0.5R_c}{N_{100}^{fail}}(|B-C|)^{kp} + \frac{0.5R_c}{N_{100}^{fail}}(|B-C|)^{kp} + \frac{0.5R_c}{N_{100}^{fail}}(|C-D|)^{kp}, \tag{4.23}$$

which simplifies to:

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-C|)^{kp} + \frac{0.5R_c}{N_{100}^{fail}}(|C-D|)^{kp}. \tag{4.24}$$

This matches the result discussed previously. Similarly, if $A < B$, then σ_1 corresponds to the state B. The cost within the step is:

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-B|)^{kp} - \frac{0.5R_c}{N_{100}^{fail}}(|B-B|)^{kp}. \tag{4.25}$$

And the cost over the three steps becomes:

$$\frac{0.5R_c}{N_{100}^{fail}}(|A-B|)^{kp} + \frac{0.5R_c}{N_{100}^{fail}}(|B-C|)^{kp} + \frac{0.5R_c}{N_{100}^{fail}}(|C-D|)^{kp}. \tag{4.26}$$

This aligns with the previously derived result.

This cost model depends on the current state $E_B(k)$ and the action, as $E_B(k+1)$ is determined by $E_B(k)$ and the corresponding action. Since the "subsequent local extrema" σ_k is also related to the state $E_B(k)$, this cost model can be integrated into a standard reward function of a POMDP. Given that standard dynamic programming (DP) is a

recursive algorithm, an additional array can be created to store the "subsequent local extrema" for all states. With a program that updates these values to calculate the aging cost at each step k, this approach can ensure that the aging cost is consistently accounted for during the optimization process.

4.3.5 Extension to Rollout Method

Since the standard DP algorithm operates recursively, the concept of "subsequent local extrema" can be applied to calculate the aging cost by counting "half-cycles." However, the rollout method is a forward simulation-based planning strategy. To calculate the aging cost during the rollout simulation, we can modify the cost model in equation 4.21 and introduce a reversed term called "prior local extrema."

Let σ_k^{pr} denote the "prior local extrema", or the previous switching point between charge and discharge actions seen from a state $E_B(k)$, the cost of battery life cycle loss in equation 4.21 can be rewritten as:

$$C_{loss}(k) = \frac{0.5R_c}{N_{100}^{fail}}\left(\left|\frac{E_B(k+1)-\sigma_{k+1}^{pr}}{E_{max}}\right|\right)^{kp} - \frac{0.5R_c}{N_{100}^{fail}}\left(\left|\frac{E_B(k)-\sigma_{k+1}^{pr}}{E_{max}}\right|\right)^{kp}. \tag{4.27}$$

To see how this cost model works in a forward planning manner, we simplify the expression of this formula as $C(k) = g\left[\left|s_{k+1}-\sigma_{k+1}^{pr}\right|\right] - g\left[\left|s_k-\sigma_{k+1}^{pr}\right|\right]$, as other parameters such as R_c, E_{max}, kp, and N_{100}^{fail} are fixed and depend on the battery specification. s_k is the state at k that is equivalent to $E_B(k)$, the energy stored in the battery at k. We use the following Figure 4.2 as a reference to verify the new cost model of battery life loss:

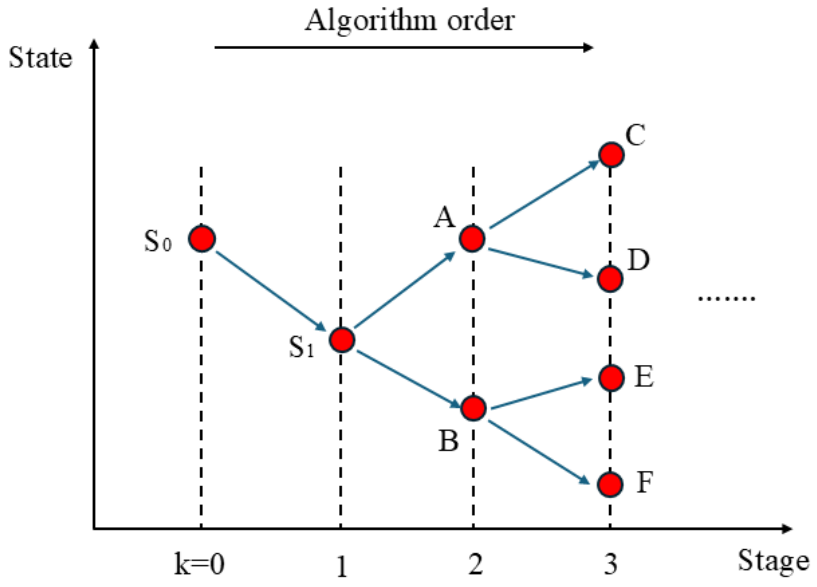

Figure 4.2: Rollout planning order.

- When $k = 0$, the "prior local extrema" σ_1^{pr} is s_0, so the cost at this stage $C_{loss}(0) = g\left[\left|s_1 - \sigma_1^{pr}\right|\right] - g\left[\left|s_0 - \sigma_1^{pr}\right|\right] = g\left[\left|s_1 - s_0\right|\right]$.
- When $k = 1$, there are two types of actions correspond to charge (A) and discharge (B). First we analyze S_2^A: the prior switching point σ_2^{pr} under this scenario is s_1, and the cost $C_{loss(A)}(1) = g\left[\left|s_2 - \sigma_2^{pr}\right|\right] - g\left[\left|s_1 - \sigma_2^{pr}\right|\right] = g\left[\left|s_2^A - s_1\right|\right] - g\left[\left|s_1 - s_1\right|\right] = g\left[\left|s_2^A - s_1\right|\right]$. The additive cost in stage 0 and 1 will be $C_{loss}(0) + C_{loss(A)}(1) = g\left[\left|s_1 - s_0\right|\right] + g\left[\left|s_2^A - s_1\right|\right]$, which is consistent with the result shown in the figure.
- On the other hand, when executing a discharge action at $k = 1$ leads to state B, the prior local extrema term σ_2^{pr} will be s_0, and the cost will become $C_{loss(B)}(1) = g\left[\left|s_2^B - s_0\right|\right] - g\left[\left|s_1 - s_0\right|\right]$. The additive cost over the two stages is $C_{loss}(0) + C_{loss(B)}(1) = g\left[\left|s_1 - s_0\right|\right] + g\left[\left|s_2^B - s_0\right|\right] - g\left[\left|s_1 - s_0\right|\right] = g\left[\left|s_2^B - s_0\right|\right]$, which also matches the result in the figure.
- At stage 2, we only verify the cost of D (discharge) and E (charge), one can easily check the other two conditions using equation 4.27, as well as decisions in subsequent stages.

- When D is the next state from stage 2, the term $\sigma_3^{pr} = s_2^A$, and the cost $C_{loss(D)}(2) = g\left[\left|s_3^D - s_2^A\right|\right] - g\left[\left|s_2^A - s_2^A\right|\right] = g\left[\left|s_3^D - s_2^A\right|\right]$. The additive cost over the three stages will be: $C_{loss}(0) + C_{loss(A)}(1) + C_{loss(D)}(2) = g\left[\left|s_1 - s_0\right|\right] + g\left[\left|s_2^A - s_1\right|\right] + g\left[\left|s_3^D - s_2^A\right|\right]$, and can be verified by the figure.

- When E is the next state from stage 2, the term $\sigma_3^{pr} = s_2^B$, and the cost $C_{loss(E)}(2) = g\left[\left|s_3^E - s_2^B\right|\right] - g\left[\left|s_2^B - s_2^B\right|\right] = g\left[\left|s_3^E - s_2^B\right|\right]$. The additive cost for the three stages will be: $C_{loss}(0) + C_{loss(B)}(1) + C_{loss(E)}(2) = g\left[\left|s_1 - s_0\right|\right] + g\left[\left|s_2^B - s_0\right|\right] - g\left[\left|s_1 - s_0\right|\right] + g\left[\left|s_3^E - s_2^B\right|\right] = g\left[\left|s_2^B - s_0\right|\right] + g\left[\left|s_3^E - s_2^B\right|\right]$, and will match the pattern in the figure.

To sum up, we modified our previous work by redefining a term named "prior local extrema" so that the aging cost model is compatible with a forward simulation planning algorithm like rollout. Similar to the method used in DP, an extra buffer to store and update this prior switching point term is required for every state and action pair, which can be easily solved. In particular, for $k \geqq 1$, $\sigma_{k+1}^{pr} \leftarrow \sigma_k^{pr}, a_k$, which means the prior local extrema at $k+1$ is determined by the prior local extrema at k and the action at k.

4.4 Summary

In this chapter, we present a smart grid scenario that could be handled by our algorithm in the previous chapter. A novel approach to incorporate a detailed battery aging model into the POMDP is presented, which can be used to optimize the operation of DERs in microgrids, including activities such as biddings of individual aggregators in the market, and the allocation of the regulation signal of each aggregator. Also, an updated battery cost model is suitable for assessing the reward function in rollout simulation and Monte Carlo tree search.

CHAPTER 5

DISCUSSION

In large-scale multi-agent planning problems, achieving computational efficiency and solution quality is a substantial challenge. Our hybrid policy, combined with a tree-based sampling strategy, seeks to balance exploration and exploitation, effectively addressing some of the inherent complexities in multi-agent systems. This section discusses the application of these methods and the challenges encountered when scaling them to large-scale problems.

5.1 Scalability Issues

As the number of agents, actions, and states increases, scalability becomes the most pressing challenge in multi-agent planning. The *curse of dimensionality* affects both the belief space and the action space:

- **Joint Action Explosion**: The exponentially growing number of joint action spaces makes brute-force exploration infeasible. The hybrid policy mitigates this by guiding action selection through one-step lookahead with rollout simulation, reducing the reliance on exhaustive search.
- **Tree-Based Sampling**: Our tree-based sampling strategy addresses scalability by selectively focusing on branches with higher uncertainty or value gaps. This

prioritization reduces the number of sampled belief nodes while preserving solution quality.

However, even with these methods, achieving scalability for hundreds of agents or extremely large state spaces remains non-trivial. The agent-by-agent approach can mitigate this issue in planning, which could be combined with extensions on distributed computation frameworks or hierarchical decomposition techniques to partition the problem into smaller, manageable subproblems.

5.2 Continuous State Discretization

In real-world applications, state spaces are mostly continuous (e.g., physical locations, velocity, or energy levels). Applying point-based solvers and tree-based sampling strategies to such problems requires *state discretization*, which introduces challenges:

- **Curse of Dimensionality in Discretization**: A fine discretization of continuous states leads to an explosion in the number of belief points. However, coarse discretization may degrade the quality of the solution.
- **Adaptive Sampling**: The tree-based sampling strategy can help adaptively refine state discretization by focusing sampling efforts on regions of the belief space that contribute most to the improvement in value function.

The hybrid policy's ability to guide decisions using approximate solutions (e.g., through a one-step lookahead and rollout) provides a pragmatic way to balance precision and computational cost in continuous domains. A potential solution is to leverage the method of state aggregation introduced in [11], to simplify the computational complexity by clustering similar states into aggregate states. This reduces the dimensionality of the state space, enabling efficient value computation and policy representation. Instead of assigning values and actions to individual states, a single value or action is assigned to each aggregate, promoting generalization while sacrificing some accuracy. The author also provided theoretical bounds for the approximation errors introduced by aggregation, emphasizing the importance of careful design to balance accuracy and computational efficiency. The method is applicable to both model-based and model-free methods, offering a practical tool for reducing complexity while retaining effective decision-making

capabilities.

5.3 Changing Dynamics

In dynamic environments where system dynamics (transition probabilities, observation models, or even action effects) evolve over time, maintaining robust policies becomes challenging. These *non-stationary dynamics* require continual adaptation:

- **Tree-Based Sampling for Dynamic Updates**: By leveraging tree-based sampling, regions of the belief space affected by changing dynamics can be revisited and updated efficiently. This ensures that the value function remains relevant in evolving environments.
- **Hybrid Policy for Replanning**: The hybrid policy, which combines one-step lookahead and rollout simulation, is particularly useful for on-line planning in changing dynamics. Instead of relying solely on offline policies, agents can adapt to changes by replanning at each decision step.

Incorporating model-learning methods into the existing framework could further enhance adaptability, especially when dynamics are unknown or partially observable.

5.4 Complex Reward Functions

Realistic multi-agent planning problems often feature *complex reward structures* with dependencies on joint states, actions, or temporal constraints. Examples include cooperative tasks where rewards are shared or adversarial settings where rewards conflict.

- **Value Function Approximation**: Complex rewards make the value function difficult to approximate, particularly in large joint belief spaces. The hybrid policy addresses this by using rollout simulations to approximate long-term rewards, leveraging the existing alpha vectors for computational efficiency.
- **Tree-Based Sampling Focus**: The sampling strategy can prioritize regions of the belief space with high reward potential or significant gaps between the lower and

upper bounds, which ensures that computational resources are concentrated on areas critical to reward maximization.

The battery aging model we proposed previously is applicable to many storage units supported DERs. However, studies on cost models of different power electronics converters will be required, as well as studies on integrating reward shaping techniques that facilitate learning in problems with sparse or delayed rewards.

5.5 Open/Dynamic Group Composition

In open or dynamic settings, agents may enter or leave the environment over time. This characteristic is common in robotic fleets, network routing, or dynamic team compositions in human-AI collaboration.

- **Policy Adaptation**: Existing policies become suboptimal when agents dynamically enter or leave the system, as the joint state and action spaces change. The hybrid policy, with its online replanning capabilities, can adapt to these changes more effectively than static offline policies.
- **Tree-Based Sampling in Dynamic Groups**: The sampling strategy can be extended to dynamically prune or expand the search tree as agents join or leave. New regions of the belief space that emerge from group changes can be prioritized for exploration.

These approaches require further development to address challenges such as ensuring fairness, synchronization, and robustness in decentralized settings.

5.6 Summary

Incorporating the hybrid policy and tree-based sampling strategy into large-scale multi-agent planning problems introduces significant advantages, particularly in addressing scalability, continuous states, dynamic environments, complex rewards, and open group compositions. By using one-step lookahead with rollout simulations and focusing exploration on high-value belief regions, these methods strike a balance between computational feasibility and solution quality. However, challenges remain, particularly in scaling

to real-world problems with hundreds of agents, evolving dynamics, and continuous state spaces. Future research should explore distributed frameworks, model learning, and adaptive sampling techniques to further enhance the applicability of these approaches.

CHAPTER 6

CONCLUSION AND FURTHER STUDY

6.1 Conclusion

In this work, we explored a decentralized multi-agent planning method and applied them within an approximate policy iteration framework to address challenging problems characterized by stochasticity, partial observability, and infinite horizons. We evaluated our approach on two representative benchmark problems: the decentralized tiger and the multi-broadcast. Based on the simulation result, we demonstrated the cost improvement property of multi-agent rollout, along with the potential superlinear convergence property from lookahead optimization. Additionally, we showed that the multi-agent planning algorithm can operate in a decentralized environment by incorporating a common information approach, which consistently improves the policy at each iteration, eventually converging to a sub-optimal policy that outperforms the base policy. Moreover, we reported promising numerical results for the decentralized scenario, further extending the applicability of our multi-agent planning methods in real-world applications. Our case study suggests that these methods are particularly effective for robotics and smart grid applications, where large teams of agents must collaboratively solve complex tasks while overcoming challenges such as stochastic dynamics, partial observability, and communication limitations.

"On-line play" could be a powerful technique to enhance RL algorithms, but enabling execution-time planning for agents in a fully decentralized environment is infeasible

due to the infinite hierarchy of belief reasoning among agents. It could be approximately achieved via the common information approach that uses knowledge from Bayesian game theory. In particular, it converts a Dec-POMDP model to a non-standard POMDP model, which assigns strategies/prescriptions to agents based on their types/action-observation histories.

The partial history sharing pattern used in the common information approach can be considered as a means of implicit communication, which allows the coordinator to construct a belief over the environment state and agents' private information. This would require additional design of information sharing protocols and modeling on disturbances, so it is necessary to assess the trade-off between improvements via on-line planning and additional costs incurred by implicit communication. In addition, we believe that this type of implicit communication could play a vital role for expanding multi-agent systems. For instance, when connecting multiple microgrids to form a network, or when multiple groups of drones are joined together to perform tasks, indirect communications could be necessary to overcome communication/operation delays, and potentially avoid a completely retraining on all agents to adapt to environments. There may be other approaches that can achieve this, such as signaling information, or certain key information sufficient to summarize the influence of one agent's policies on another, so others can choose their best responses as in game-theoretic approaches.

6.2 Future Direction of Research

A core aspect of RL involves agents that interact with an environment through actions to learn optimal policies. However, most studies focus on interactions with simulations and off-line computations rather than real-world environments, as the cost of testing off-line trained agents in complex, on-line systems is often prohibitive. As a result, researchers tend to focus on developing RL algorithms tailored to excel in specific simulation-based tasks, which often restricts their adaptability to new environments and real-world scenarios. In the context of microgrid control, numerous RL-based algorithms have been proposed for various grid models. However, unlike benchmark problems, practical microgrid control involves dynamic environments in which parameters such as electricity prices, renewable generation forecasts, and load demands fluctuate over time. These changes require real-time updates of probability distributions and fast on-line replanning

under a model predictive control framework. Many existing RL algorithms, designed and trained for fixed state-transition probabilities in simulated environments, struggle to adapt to the evolving conditions of real-world microgrid operations.

Another challenge arises in MARL when the number of agents changes, as seen in residential microgrids where the integration of distributed storage units and renewable resources is increasing. For instance, installing solar panels and small battery energy storage systems (BESSs) can turn previously uncontrollable nodes into controllable agents. This shift requires modifying and retraining the MARL algorithm, leading to costly adjustments in policy or value function neural networks. Even small changes in the grid can require significant tuning of the control framework.

Addressing these challenges requires a shift toward meta-RL algorithms capable of adapting to diverse tasks and real-world changes. Instead of focusing on fixed task control schemes for microgrids, there is a growing need to develop meta-RL approaches that can handle varying environments, evolving grid configurations, and dynamic operational requirements effectively and efficiently.

APPENDIX

Consider a standard cooperative multi-agent environment such as a Dec-POMDP model, and assume agents share all their observations and actions during the centralized training phase, a belief of agents is a probability distribution over states:

A.1 Belief State as Sufficient Statistic

$$b(s_t) = \mathbb{P}(s_t \mid \mathbf{o}_t, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}, \cdots, \mathbf{a}_1, \mathbf{o}_1, \mathbf{a}_0) \tag{A.1}$$

where $\mathbf{o}$ and $\mathbf{a}$ indicate the joint actions and observations; The transition functions $\mathbb{P}(s' \mid s, \mathbf{a})$, and $\mathbb{P}(\mathbf{o} \mid s', \mathbf{a})$ are also provided. Define $I_t = (\mathbf{o}_t, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}, \cdots, \mathbf{a}_1, \mathbf{o}_1, \mathbf{a}_0)$ as the information vector up to t, which has a state space of expanding dimension. The belief is a sufficient statistic for the problem that would summarize all the essential content of I_t and would (ideally) have a smaller dimension. To proof this, let

$$\begin{aligned} \mathbb{P}(s_t, \mathbf{o}_t \mid I_t, \mathbf{a}_t) &= \mathbb{P}(s_t \mid I_t, \mathbf{a}_t) \times \mathbb{P}(\mathbf{o}_t \mid s_t, I_t, \mathbf{a}_t) \\ &= \mathbb{P}(s_t \mid I_t) \times \mathbb{P}(\mathbf{o}_t \mid s_t, \mathbf{a}_t) \end{aligned} \tag{A.2}$$

The first equality follows the definition of conditional probability and its product rule. The second equality follows since s_t does not depend on $\mathbf{a}_t$, and $\mathbf{o}_t$ depends only on s_t and $\mathbf{a}_t$ according to the transition functions. Since $\mathbb{P}(\mathbf{o}_t \mid s_t, \mathbf{a}_t)$ is problem given, the expected immediate reward $r(s, \mathbf{a})$ can be computed as

$$\mathbb{E}_{s_t,\mathbf{o}_t}[r(s_t,\mathbf{a}_t) \mid I_t,\mathbf{a}_t] = r(\mathbb{P}(s_t \mid I_t),\mathbf{a}_t) = r(b(s_t),\mathbf{a}_t)) \tag{A.3}$$

So the belief $b(s_t)$ is a sufficient statistic to summarize the information vector I_t, see chapter 4 of [7] for more detail.

A.2 Belief Update Formula

Next, we show how to derive the belief update formula via Baye's Theorem. The posterior probability of being in state s' given the joint observation $\mathbf{o}$ the joint action $\mathbf{a}$ is:

$$b'(s') = \mathbb{P}(s' \mid \mathbf{o},\mathbf{a},b) \tag{A.4}$$

Using Baye's Theorem:

$$\mathbb{P}(s' \mid \mathbf{o},\mathbf{a},b) = \frac{\mathbb{P}(\mathbf{o} \mid s',\mathbf{a},b)\mathbb{P}(s' \mid \mathbf{a},b)}{\mathbb{P}(\mathbf{o} \mid \mathbf{a},b)} \tag{A.5}$$

Notice that $\mathbb{P}(\mathbf{o} \mid s',\mathbf{a},b) = \mathbb{P}(\mathbf{o} \mid s',\mathbf{a})$ as the joint observation $\mathbf{o}$ depends only on state s' and the joint action $\mathbf{a}$, according to the observation probability model; the state transition model $\mathbb{P}(s' \mid \mathbf{a},b) = \sum_{s\in S}\mathbb{P}(s' \mid s,\mathbf{a})b(s)$, where $b(s)$ is the prior belief and $\mathbb{P}(s' \mid s,\mathbf{a})$ is the state transition probability. The denominator term $\mathbb{P}(\mathbf{o} \mid \mathbf{a},b)$ is a normalizing constant, and it is given as:

$$\begin{aligned}\mathbb{P}(\mathbf{o} \mid \mathbf{a},b) &= \sum_{s'\in S}\mathbb{P}(\mathbf{o} \mid s',\mathbf{a},b)\mathbb{P}(s' \mid \mathbf{a},b) \\ &= \sum_{s'\in S}\mathbb{P}(\mathbf{o} \mid s',\mathbf{a})\sum_{s\in S}\mathbb{P}(s' \mid s,\mathbf{a})b(s)\end{aligned} \tag{A.6}$$

to ensure the next belief $b'(s')$ is a valid probability distribution(i.e non-negative and probabilities sum to 1, $\sum_{s'} b'(s') = 1$).

Therefore the belief update formula is:

$$b'(s') = \frac{\mathbb{P}(\mathbf{o} \mid s',\mathbf{a})\sum_s \mathbb{P}(s' \mid s,\mathbf{a})b(s)}{\sum_{s'}\mathbb{P}(\mathbf{o} \mid s',\mathbf{a})\sum_s \mathbb{P}(s' \mid s,\mathbf{a})b(s)} \tag{A.7}$$

A.3 Proof: The Value Function of a Multi-Agent POMDP is Piecewise Linear and Convex (PWLC)

1. Bellman Equation for Multi-Agent POMDPs

The value function for a multi-agent POMDP at belief state b is given by:

$$V_{t+1}(b) = \max_{a \in A} \left[R(b,a) + \gamma \sum_{o \in O} \Pr(o \mid b,a) V_t(b') \right],$$

where:

- b: Belief over the joint state space S,
- $a \in A$: Joint action space for all agents,
- $o \in O$: Joint observation space,
- $\Pr(o \mid b,a)$: Probability of observing o after action a in belief b,
- $R(b,a)$: Expected immediate reward for action a in belief b,
- b': Updated belief after observing o,
- γ: Discount factor, $\gamma \in [0,1)$.

2. Horizon 1 Value Function

At horizon $t = 1$, the value function $V_1(b)$ is defined as:

$$V_1(b) = \max_{a \in A} R(b,a).$$

The immediate reward $R(b,a)$ is linear in b:

$$R(b,a) = \sum_{s \in S} b(s) r(s,a),$$

where $r(s,a)$ is the reward for being in state s and taking action a. Since the maximum of linear functions is PWLC, $V_1(b)$ is PWLC.

3. Inductive Hypothesis

Assume that at horizon t, the value function $V_t(b)$ is PWLC:

$$V_t(b) = \max_{\alpha \in \Gamma_t} b \cdot \alpha,$$

where Γ_t is a finite set of alpha-vectors and $b \cdot \alpha$ is the dot product of the belief b and the alpha-vector α.

4. Bellman Backup Preserves PWLC

The Bellman backup is:

$$V_{t+1}(b) = \max_{a \in A} \left[R(b,a) + \gamma \sum_{o \in O} \Pr(o \mid b,a) V_t(b') \right].$$

Step 1: Immediate Reward is Linear. The immediate reward $R(b,a)$ is linear:

$$R(b,a) = \sum_{s \in S} b(s) r(s,a).$$

Step 2: Belief Update. The updated belief b' after action a and observation o is:

$$b'(s') = \frac{\Pr(o \mid s', a) \sum_{s \in S} \Pr(s' \mid s, a) b(s)}{\Pr(o \mid b, a)},$$

where:

$$\Pr(o \mid b,a) = \sum_{s' \in S} \Pr(o \mid s', a) \sum_{s \in S} \Pr(s' \mid s, a) b(s).$$

Step 3: Substituting $V_t(b')$. By the inductive hypothesis:

$$V_t(b') = \max_{\alpha \in \Gamma_t} b' \cdot \alpha.$$

Substitute b' into $V_t(b')$:

$$\sum_{o \in O} \Pr(o \mid b,a) V_t(b') = \sum_{o \in O} \Pr(o \mid b,a) \max_{\alpha \in \Gamma_t} b \cdot \alpha_o,$$

where α_o accounts for the belief update b'.

Step 4: Maximizing over Actions. Finally, we compute:

$$V_{t+1}(b) = \max_{a \in A} Q(b,a),$$

where:

$$Q(b,a) = R(b,a) + \gamma \sum_{o \in O} \Pr(o \mid b,a) \max_{\alpha \in \Gamma_t} b \cdot \alpha_o.$$

Since $Q(b,a)$ is the maximum of linear functions, $V_{t+1}(b)$ remains PWLC.

5. Conclusion

By induction, the value function $V_t(b)$ of a multi-agent POMDP is piecewise linear and convex:

$$V_t(b) = \max_{\alpha \in \Gamma_t} b \cdot \alpha.$$

BIBLIOGRAPHY

[1] S. V. ALBRECHT, F. CHRISTIANOS, AND L. SCHÄFER, *Multi-agent reinforcement learning: Foundations and modern approaches*, MIT Press, 2024.

[2] C. AMATO AND F. OLIEHOEK, *Scalable planning and learning for multiagent pomdps*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, 2015.

[3] N. BARD, J. N. FOERSTER, S. CHANDAR, N. BURCH, M. LANCTOT, H. F. SONG, E. PARISOTTO, V. DUMOULIN, S. MOITRA, E. HUGHES, ET AL., *The hanabi challenge: A new frontier for ai research*, Artificial Intelligence, 280 (2020), p. 103216.

[4] M. G. BELLEMARE, W. DABNEY, AND R. MUNOS, *A distributional perspective on reinforcement learning*, in International conference on machine learning, PMLR, 2017, pp. 449–458.

[5] D. S. BERNSTEIN, C. AMATO, E. A. HANSEN, AND S. ZILBERSTEIN, *Policy iteration for decentralized control of markov decision processes*, Journal of Artificial Intelligence Research, 34 (2009), pp. 89–132.

[6] D. S. BERNSTEIN, R. GIVAN, N. IMMERMAN, AND S. ZILBERSTEIN, *The complexity of decentralized control of markov decision processes*, Mathematics of operations research, 27 (2002), pp. 819–840.

[7] D. BERTSEKAS, *Dynamic programming and optimal control: Volume I*, vol. 4, Athena scientific, 2012.

[8] ——, *Multiagent reinforcement learning: Rollout and policy iteration*, IEEE/CAA Journal of Automatica Sinica, 8 (2021), pp. 249–272.

[9] ——, *Lessons from AlphaZero for optimal, model predictive, and adaptive control*, Athena Scientific, 2022.

[10] ——, *Lessons from AlphaZero for optimal, model predictive, and adaptive control*, Athena Scientific, 2022.

[11] ——, *A course in reinforcement learning*, Athena Scientific, 2023.

[12] S. BHATTACHARYA, S. KAILAS, S. BADYAL, S. GIL, AND D. BERTSEKAS, *Multiagent reinforcement learning: Rollout and policy iteration for pomdp with application to multi-robot problems*, IEEE Transactions on Robotics, (2023).

[13] T. CHEN, S. BU, X. LIU, J. KANG, F. R. YU, AND Z. HAN, *Peer-to-peer energy trading and energy conversion in interconnected multi-energy microgrids using multi-agent deep reinforcement learning*, IEEE transactions on smart grid, 13 (2021), pp. 715–727.

[14] P.-A. COQUELIN AND R. MUNOS, *Bandit algorithms for tree search*, in Uncertainty in Artificial Intelligence, 2007.

[15] R. COULOM, *Efficient selectivity and backup operators in monte-carlo tree search*, in International conference on computers and games, Springer, 2006, pp. 72–83.

[16] X. FANG, J. WANG, G. SONG, Y. HAN, Q. ZHAO, AND Z. CAO, *Multi-agent reinforcement learning approach for residential microgrid energy scheduling*, Energies, 13 (2019), p. 123.

[17] J. FOERSTER, G. FARQUHAR, T. AFOURAS, N. NARDELLI, AND S. WHITESON, *Counterfactual multi-agent policy gradients*, in Proceedings of the AAAI conference on artificial intelligence, vol. 32, 2018.

[18] J. FOERSTER, F. SONG, E. HUGHES, N. BURCH, I. DUNNING, S. WHITESON, M. BOTVINICK, AND M. BOWLING, *Bayesian action decoder for deep multi-agent reinforcement learning*, in International Conference on Machine Learning, PMLR, 2019, pp. 1942–1951.

[19] T. HAARNOJA, A. ZHOU, P. ABBEEL, AND S. LEVINE, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, in International conference on machine learning, PMLR, 2018, pp. 1861–1870.

[20] R. HABIBIFAR, A. A. LEKVAN, AND M. EHSAN, *A risk-constrained decision support tool for ev aggregators participating in energy and frequency regulation markets*, Electric Power Systems Research, 185 (2020), p. 106367.

[21] G. HE, Q. CHEN, C. KANG, P. PINSON, AND Q. XIA, *Optimal bidding strategy of battery storage in power markets considering performance-based regulation and battery cycle life*, IEEE Transactions on Smart Grid, 7 (2015), pp. 2359–2367.

[22] JOSH ACHIAM, *Part 2: Kinds of RL algorithms*. `https://spinningup.openai.com/en/latest/index.html`, 2018.

[23] L. P. KAELBLING, *Learning in embedded systems*, MIT press, 1993.

[24] H. KURNIAWATI, D. HSU, AND W. S. LEE, *SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces*, MIT Press, 2009, pp. 65–72.

[25] F.-D. LI, M. WU, Y. HE, AND X. CHEN, *Optimal control in microgrid using multi-agent reinforcement learning*, ISA transactions, 51 (2012), pp. 743–751.

[26] J. LI, S. YANG, AND T. YU, *Data-driven cooperative load frequency control method for microgrids using effective exploration-distributed multi-agent deep reinforcement learning*, IET Renewable Power Generation, 16 (2022), pp. 655–670.

[27] T. LILLICRAP, *Continuous control with deep reinforcement learning*, arXiv preprint arXiv:1509.02971, (2015).

[28] R. LOWE, Y. I. WU, A. TAMAR, J. HARB, O. PIETER ABBEEL, AND I. MORDATCH, *Multi-agent actor-critic for mixed cooperative-competitive environments*, Advances in neural information processing systems, 30 (2017).

[29] A. MAHAJAN, *Optimal decentralized control of coupled subsystems with control sharing*, IEEE Transactions on Automatic Control, 58 (2013), pp. 2377–2382.

[30] A. MAHAJAN, T. RASHID, M. SAMVELYAN, AND S. WHITESON, *Maven: Multi-agent variational exploration*, Advances in neural information processing systems, 32 (2019).

[31] V. MNIH, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602, (2013).

[32] V. MNIH, A. P. BADIA, M. MIRZA, A. GRAVES, T. HARLEY, T. P. LILLICRAP, D. SILVER, AND K. KAVUKCUOGLU, *Asynchronous methods for deep reinforcement learning*, in Proceedings of the 33rd International Conference on Interna-

tional Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, pp. 1928–1937.

[33] M. S. MUNIR, S. F. ABEDIN, N. H. TRAN, Z. HAN, E.-N. HUH, AND C. S. HONG, *Risk-aware energy scheduling for edge computing with microgrid: A multi-agent deep reinforcement learning approach*, IEEE Transactions on Network and Service Management, 18 (2021), pp. 3476–3497.

[34] R. NAIR, M. TAMBE, M. YOKOO, D. PYNADATH, AND S. MARSELLA, *Taming decentralized pomdps: Towards efficient policy computation for multiagent settings*, in IJCAI, vol. 3, 2003, pp. 705–711.

[35] A. NAYYAR, A. MAHAJAN, AND D. TENEKETZIS, *Decentralized stochastic control with partial history sharing: A common information approach*, IEEE Transactions on Automatic Control, 58 (2013), pp. 1644–1658.

[36] H. NIE, Y. CHEN, Y. XIA, S. HUANG, AND B. LIU, *Optimizing the post-disaster control of islanded microgrid: A multi-agent deep reinforcement learning approach*, IEEE Access, 8 (2020), pp. 153455–153469.

[37] H. K. NUNNA, A. SESETTI, A. K. RATHORE, AND S. DOOLLA, *Multiagent-based energy trading platform for energy storage systems in distribution systems with interconnected microgrids*, IEEE Transactions on Industry Applications, 56 (2020), pp. 3207–3217.

[38] F. A. OLIEHOEK, C. AMATO, ET AL., *A concise introduction to decentralized POMDPs*, vol. 1, Springer, 2016.

[39] A. OSHNOEI, M. KHERADMANDI, F. BLAABJERG, N. D. HATZIARGYRIOU, S. MUYEEN, AND A. ANVARI-MOGHADDAM, *Coordinated control scheme for provision of frequency regulation service by virtual power plants*, Applied Energy, 325 (2022), p. 119734.

[40] Y. OUYANG, H. TAVAFOGHI, AND D. TENEKETZIS, *Dynamic games with asymmetric information: Common information based perfect bayesian equilibria and sequential decomposition*, IEEE Transactions on Automatic Control, 62 (2016), pp. 222–237.

[41] J. PAJARINEN AND J. PELTONEN, *Periodic finite state controllers for efficient pomdp and dec-pomdp planning*, Advances in neural information processing systems, 24 (2011).

[42] B. PENG, T. RASHID, C. SCHROEDER DE WITT, P.-A. KAMIENNY, P. TORR, W. BÖHMER, AND S. WHITESON, *Facmac: Factored multi-agent centralised policy gradients*, Advances in Neural Information Processing Systems, 34 (2021), pp. 12208–12221.

[43] T. PU, X. WANG, Y. CAO, Z. LIU, C. QIU, J. QIAO, AND S. ZHANG, *Power flow adjustment for smart microgrid based on edge computing and multi-agent deep reinforcement learning*, Journal of Cloud Computing, 10 (2021), pp. 1–13.

[44] M. A. RAHMAN, N. HOPNER, F. CHRISTIANOS, AND S. V. ALBRECHT, *Towards open ad hoc teamwork using graph-based policy learning*, (2021), pp. 8776–8786.

[45] T. RASHID, M. SAMVELYAN, C. S. DE WITT, G. FARQUHAR, J. FOERSTER, AND S. WHITESON, *Monotonic value function factorisation for deep multi-agent reinforcement learning*, Journal of Machine Learning Research, 21 (2020), pp. 1–51.

[46] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, arXiv preprint arXiv:1707.06347, (2017).

[47] S. SEUKEN AND S. ZILBERSTEIN, *Memory-bounded dynamic programming for dec-pomdps.*, in IJCAI, 2007, pp. 2009–2015.

[48] G. SHANI, J. PINEAU, AND R. KAPLOW, *A survey of point-based pomdp solvers*, Autonomous Agents and Multi-Agent Systems, 27 (2013), pp. 1–51.

[49] D. SILVER, T. HUBERT, J. SCHRITTWIESER, I. ANTONOGLOU, M. LAI, A. GUEZ, M. LANCTOT, L. SIFRE, D. KUMARAN, T. GRAEPEL, ET AL., *A general reinforcement learning algorithm that masters chess, shogi, and go through self-play*, Science, 362 (2018), pp. 1140–1144.

[50] T. SMITH AND R. SIMMONS, *Heuristic search value iteration for pomdps*, Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, (2012), pp. 520–527.

[51] ——, *Point-based pomdp algorithms: Improved analysis and implementation*, Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence, (2012), pp. 542–549.

[52] E. J. SONDIK, *The optimal control of partially observable markov processes over the infinite horizon: Discounted costs*, Operations research, 26 (1978), pp. 282–304.

[53] P. SRIVASTAVA, C.-Y. CHANG, AND J. CORTÉS, *Enabling der participation in frequency regulation markets*, IEEE Transactions on Control Systems Technology, 30 (2022), pp. 2391–2405.

[54] J. SUBRAMANIAN, A. SINHA, R. SERAJ, AND A. MAHAJAN, *Approximate information state for approximate planning and reinforcement learning in partially observed systems*, Journal of Machine Learning Research, 23 (2022), pp. 1–83.

[55] L. SUN, J. QIU, X. HAN, X. YIN, AND Z. DONG, *Per-use-share rental strategy of distributed bess in joint energy and frequency control ancillary services markets*, Applied Energy, 277 (2020), p. 115589.

[56] Y. SUN, Z. CAI, Z. ZHANG, C. GUO, G. MA, AND Y. HAN, *Coordinated energy scheduling of a distributed multi-microgrid system based on multi-agent decisions*, Energies, 13 (2020), p. 4077.

[57] P. SUNEHAG, G. LEVER, A. GRUSLYS, W. M. CZARNECKI, V. ZAMBALDI, M. JADERBERG, M. LANCTOT, N. SONNERAT, J. Z. LEIBO, K. TUYLS, AND T. GRAEPEL, *Value-decomposition networks for cooperative multi-agent learning based on team reward*, in Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, Richland, SC, 2018, International Foundation for Autonomous Agents and Multiagent Systems, p. 2085,Äì2087.

[58] R. S. SUTTON AND A. G. BARTO, *Reinforcement learning: An introduction*, MIT press, 2018.

[59] R. S. SUTTON, D. MCALLESTER, S. SINGH, AND Y. MANSOUR, *Policy gradient methods for reinforcement learning with function approximation*, Advances in neural information processing systems, 12 (1999).

[60] D. TANG, A. NAYYAR, AND R. JAIN, *A novel point-based algorithm for multi-agent control using the common information approach*, in 2023 62nd IEEE Conference on Decision and Control (CDC), IEEE, 2023, pp. 1432–1437.

[61] H. TAVAFOGHI, Y. OUYANG, AND D. TENEKETZIS, *A unified approach to dynamic decision problems with asymmetric information: Nonstrategic agents*, IEEE Transactions on Automatic Control, 67 (2021), pp. 1105–1119.

[62] G. TESAURO AND G. GALPERIN, *On-line policy improvement using monte-carlo search*, Advances in neural information processing systems, 9 (1996).

[63] Y. XIE, W. GUO, Q. WU, AND K. WANG, *Robust mpc-based bidding strategy for wind storage systems in real-time energy and regulation markets*, International Journal of Electrical Power & Energy Systems, 124 (2021), p. 106361.

[64] K. ZHANG, Z. YANG, AND T. BAŞAR, *Multi-agent reinforcement learning: A selective overview of theories and algorithms*, Handbook of reinforcement learning and control, (2021), pp. 321–384.

[65] H. ZHOU, A. ARAL, I. BRANDIĆ, AND M. EROL-KANTARCI, *Multiagent bayesian deep reinforcement learning for microgrid energy management under communication failures*, IEEE Internet of Things Journal, 9 (2021), pp. 11685–11698.

[66] J. ZHOU, S. LUO, AND F. CHEN, *Effects of personality traits on user trust in human–machine collaborations*, Journal on Multimodal User Interfaces, 14 (2020), pp. 387–400.

[67] W. ZHUO AND A. V. SAVKIN, *Profit maximizing control of a microgrid with renewable generation and bess based on a battery cycle life model and energy price forecasting*, Energies, 12 (2019), p. 2904.