Applied Energy 318 (2022) 119151

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

Renewable energy integration and microgrid energy trading using multi-agent deep reinforcement learning

Daniel J.B. Harrold [a,*], Jun Cao [b], Zhong Fan [a]

[a] School of Computing and Mathematics, Keele University, Keele, United Kingdom
[b] Environmental Research and Innovation Department, Sustainable Energy Systems Group, Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg

ARTICLE INFO

Keywords:
Deep deterministic policy gradients
Energy arbitrage
Hybrid energy storage systems
Multi-agent systems
Renewable energy

ABSTRACT

To reduce global greenhouse gas emissions, the world must find intelligent solutions to maximise the utilisation of carbon-free renewable energy sources. In this paper, multi-agent reinforcement learning is used to control a microgrid in a mixed cooperative and competitive setting. The agents observe fluctuating energy demand, dynamic wholesale energy prices, and intermittent renewable energy sources to control a hybrid energy storage system to maximise the utilisation of the renewables to reduce the energy costs of the grid. In addition, an aggregator agent trades with external microgrids competing against one another and the aggregator to reduce their own energy bills. For this, the algorithm deep deterministic policy gradients (DDPG) and multi-agent DDPG (MADDPG) are used to compare the use of a single global controller versus multiple distributed agents, along with the single and multi-agent variants of distributional DDPG (D3PG) and twin delayed DDPG (TD3). The research found it is significantly more profitable for the primary microgrid to sell energy on its own terms rather than selling back to the utility grid, and is also beneficial for the external microgrids as they also reduce their own energy bills. The methods that produced the greatest profits were the multi-agent approaches where each agent has its own reward function based on the principle of marginal contribution from game theory. The multi-agent approaches were better able to evaluate their performance controlling individual components of the environment which allowed them to develop their own unique policies for the different types of energy storage system.

1. Introduction

The energy sector is responsible for the overwhelming majority of global greenhouse gas emissions [1]. As the world looks to become more sustainable, a key component of reducing emissions is by moving away from traditional energy generation by increasing the penetration of renewable energy sources (RES) [2]. Although solar photovoltaic (PV) and onshore wind turbines (WT) can be amongst the most cost effective ways of providing energy [3] with the price of both decreasing greatly over the past decades [4], the intermittent and unpredictable nature of RES remains a significant barrier when compared to unsustainable but reliable coal and natural gas power.

Smart energy networks look at tackling this by more intelligently managing the supply and demand of energy. For example, energy storage systems (ESS) may be used to store energy generated from RES when there is a surplus of generation so that it may be used later at peak times. This gives the grid additional stability and security in systems with a large amount of RES [5]. A hybrid energy storage system (HESS) could also be used in which multiple ESSs are present in the same grid. This can be done to complement the characteristics of the different types, such as pairing a supercapacitor (SC) for short-term storage with a lithium-ion battery (LIB) or fuel cell for longer-term storage [6].

Although this is difficult to achieve at utility grid scales, it can be implemented more easily at smaller scales. Microgrids are local clusters of loads and distributed generation which can either be connected to the main utility grid or act independently as an island. The microgrid may want to use its ESSs for a number of reasons, including RES integration or energy arbitrage, to reduce its dependency on the utility grid [7]. Therefore, a control method is required to be able to learn behaviours for each type of ESS in a complex grid, as well as taking into account fluctuating RES generation and volatile dynamic energy prices.

For this control system, we propose reinforcement learning (RL), a branch of machine learning in which an agent learns to interact with its environment to find an optimal control policy [8]. The agent iteratively

* Corresponding author.
E-mail address: d.j.b.harrold@keele.ac.uk (D.J.B. Harrold).

https://doi.org/10.1016/j.apenergy.2022.119151
Received 4 December 2021; Received in revised form 25 March 2022; Accepted 14 April 2022
Available online 11 May 2022

Nomenclature

Abbreviations

ANN	artificial neural network
DDPG	deep deterministic policy gradients
DQN	deep Q-networks
ESS	energy storage system
HESS	hybrid ESS
LIB	lithium-ion battery
MAS	multi-agent system
MGA	microgrid aggregator
PV	photovoltaic
RES	renewable energy source
RL	reinforcement learning
SC	supercapacitor
TD3	twin delayed DDPG
VRB	vanadium redox battery
WT	wind turbine
xMG	external microgrid

Environment Parameter Symbols

x	ESS power
X	power from demand or generation
c	ESS charge
C	ESS capacity
P	energy price
η	efficiency

Reinforcement Learning Symbols

s	state
a	action
r	reward
γ	discount factor
$Q(s,a)$	critic action-value estimate
$\mu(s)$	actor policy action
θ	critic network weights
ϕ	actor network weights

chooses an action based on its observations of the environment to receive a reward, with the goal to maximise its total future reward. In addition, artificial neural networks (ANN) can be used to combine RL with deep learning for solutions to more complex problems. Model-free RL algorithms are incredibly versatile and easy to implement as they can be used in situations where there is little information or data available on the environment, as the agent learns from its own experiences through trial-and-error.

In this energy network context, the problem can also be considered as a multi-agent system (MAS) in which multiple intelligent decision-making agents must cooperate, coordinate, and negotiate with each other to achieve their individual goals [9]. In theory, RL should be a suitable learning mechanism for these agents as it too is used for sequential decision making. However, there are a number of fundamental challenges when implementing RL for a MAS such as a non-stationary environment, communication between agents, and assigning appropriate rewards for agent learning [10].

Therefore, there is a dilemma if a microgrid energy system should be managed by a single global controller or by multiple distributed agents in a MAS. In this paper, this problem is investigated using two case studies: the first controlling the HESS to maximise the utilisation of RES under dynamic pricing to reduce the grid's energy bill, and the second building on the first using an aggregator trading with other neighbouring microgrids looking to reduce their own bills.

1.1. Related literature

There is extensive research into the use of RL for ESS control to maximise RES [11], and research into the field has increased substantially over the last decade [12]. Kuznetsova et al. [13] were the first authors to use RL to control an ESS in the presence of RES, using a battery to maximise the utilisation of WT generation. However, the algorithm used is considered outdated by current standards of deep RL approaches which were developed several years after.

There is also work on the use of RL for HESS, but it is generally less explored and typically uses a single control agent to manage the multiple ESSs. Francois-Lavet et al. [14] used deep RL to control both a LIB and a hydrogen fuel cell to maximise RES utilisation in a solar microgrid. Qiu et al. [15] used RL to control both a lead–acid and vanadium redox battery (VRB) to reduce grid power losses, again in a solar microgrid. However, as both of these papers only consider two different ESSs, the scalability of these approaches when used in larger microgrids with more ESSs or other types of agents is unclear. Zhang et al. [16] used a more advanced deep RL algorithm for the control of a microgrid completed with a HESS and both PV and WT generation. The authors test a variety of different single-agent approaches for control of the entire grid, but also suggest exploring the use of a MAS as a potential future research avenue.

There are alternative model-based methods for ESS control. Prodan et al. [17] compared the use of a model predictive controller to the research of Kuznetsova et al. [13] and note better performance but requires significantly more computation. Li et al. [18] use distributed coordinated power control for voltage and frequency control using a battery, SC, and fuel cell in a combined AC/DC microgrid. However, these methods rely on having a complete model available of the environment and their performance is only as accurate as the model itself. In contrast, a RL algorithm could be trained on a digital model but would continue to learn and improve when applied to and evaluated on a real system.

Multi-agent RL has also been used for HESS and larger microgrids that would be difficult to control using a single agent. Foruzan et al. [19] uses RL to control multiple agents in a microgrid including an ESS, RES, a generator, and self-interested customers each with their own reward function. However, the algorithm used is fairly basic and restricts control to a discrete state and action space. Kofinas et al. [20] used the same algorithm but adapted with fuzzy logic to adapt the method for continuous state and action spaces in a MAS. However, more advanced methods have been developed since that can achieve this without the fuzzy logic. Mbuwir et al. [21] uses multi-agent deep RL to control a battery and heat pump, but the microgrid is much smaller with fewer agents than the two aforementioned works and is restricted to a discrete action space. Wang et al. [22] uses another multi-agent deep RL algorithm for the routing and scheduling of mobile ESSs, such as electric vehicles, in a microgrid with both PV and distribute diesel generates to reduce load shedding cost.

There are a number of papers in the past couple of years which use MADDPG papers for RES integration. Li et al. [23] use MADDPG to control RES generation power for frequency control in an integrated energy system, considering both PV and WT generation. However, these papers tackle different problems so do not consider dynamic pricing schemes. Xu et al. [24] use MADDPG for peer-to-peer energy trading between microgrids while Li et al. [25] use the algorithm for grid edge control of ESSs across different households. These papers consider a deterministic time-of-use dynamic pricing scheme that only incentives customers to use energy away from peak periods. Therefore, it does not consider more volatile and unpredictable wholesale energy prices which the agents must learn to traverse. All four papers also only consider either a cooperative setting with agents working together, or a competitive setting where selfish agents are competing against one another. MADDPG provides the flexibility to consider a mixed scenario in a MAS.

To the best of our knowledge, there is no literature on the use of multi-agent RL for ESS control considering RES integration under a wholesale energy price market. Likewise, there is no literature considering an environment in a mixed cooperative and competitive setting.

1.2. Contributions

The contributions of this paper are as follows:

- Presents a novel use case of MADDPG in a mixed cooperative and competitive MAS microgrid scenario for both RES integration and energy arbitrage. The centralised learning and decentralised execution of the method allows each agent is able to evaluate their policy by observing the actions of all other agents without having to share parameters or critic networks. This is ideal for the HESS working cooperatively as well as for self-interested agents looking to do the same for external microgrids.
- Considers both energy arbitrage and RES integration in a single problem within the MAS. An asymmetric wholesale energy market real-time pricing scheme means that the agents must respond to both the fluctuating prices and RES output to optimally reduce the microgrid's energy bills.
- Evaluates the use of a single-agent global controller versus multiple distributed agents for the control of different components of the microgrid. The individual reward function that each agent receives from the individual critic networks under MADDPG uses the concept of marginal contribution to better assess how the agents' actions impacted the joint goal of reducing energy costs. This also allows each agent in the HESS to develop their own unique policy to best suit that ESS type.

1.3. Structure

The rest of this paper is organised into the microgrid environment description in Section 2, the background into RL and the specific algorithms in Section 3, outline of the two case studies in Section 4, results and discussion in Section 5, with final conclusions made in Section 6.

2. Microgrid description

Microgrids are localised energy networks with their own demand and RES that are able to trade energy with the main utility grid or operate as an independent island [26]. This section will describe the primary microgrid used in the future case studies and how it is modelled as a Markov decision process (MDP) for RL. The first case will look solely at using the HESS to maximise the utilisation of the RES to reduce the primary microgrid's energy bills, and the second will consider using an aggregator to sell energy to neighbouring external microgrids (xMG) competing against each other to reduce their own individual energy bills.

2.1. Energy network description

The primary microgrid is fitted with a HESS as well as both PV and WT generation. It is connected to the main utility grid that sets dynamic energy prices from which the microgrid can import energy from, or sell back to at a fixed feed-in tariff. AC and DC power lines are present connected via inverters, as well as transformers between the primary microgrid and both the utility grid and WT. In addition, the primary microgrid is also connected to five xMGs via another transformer, with those xMGs are connected themselves to the main utility grid but not the wholesale energy market. The basic schematic of the environment is shown in Fig. 1.

2.1.1. ESS

The primary microgrid consists of a HESS with three different types of ESS: a LIB, a VRB, and a SC which are suited for mid-term, long-term, and short-term energy storage respectively. All three have the same maximum capacity and maximum power, shown in Fig. 1, and can transfer energy between each other without the need to go through an inverter. Some general observations of each are:

- LIB is cheapest per capacity with a low self-discharge and a good round-trip efficiency but has the fewest total lifecycles making it ideal for medium-term storage.
- VRB has a negligible self-discharge but a poor round-trip efficiency making it suited for storing energy over much longer periods of time.
- SC has a much largest number lifecycles making it by far the cheapest to operate, but a poor self-discharge makes it only ideal for dealing with fluctuations over the course of a single day.

The control policy that the agent must learn should optimally manage the contrasting characteristics of the ESS types to maximise each of their strengths and mitigate against their respective flaws.

2.1.2. Demand and renewable energy sources

The demand data collected at Keele University Campus ranges from 00:00 January 1st 2014 up to and including 23:00 December 31st 2017. The raw data is separated into different residential, industrial, and commercial sites as well as readings for key individual buildings. The demand used for this simulation is from the three main incomer substations into the campus with the half-hourly readings summed to match the frequency of the hourly weather data.

This microgrid considers both PV and WT generation with the output simulated using weather data collected at the Keele University weather station, and therefore at the same site as the demand data and over the same period of time.

The grid components are also connected using inverters between Ac and DC lines, as well as transformers to step-up or step-down voltages. Transformers connect the primary microgrid to the utility grid, the WT, and external microgrids (xMG) in the second case study. Inverters connect the main AC line of the microgrid to the HESS and PV generation which all use DC. All loads are AC.

2.1.3. Wholesale energy pricing

The microgrid is connected to the main utility grid, but can also purchase energy from a wholesale energy market under a dynamic pricing scheme. Here, the microgrid can purchase using real-time pricing set by the market operator, but can only sell back to the utility grid at a constant feed-in tariff.

By reacting to the uneven dynamic pricing scheme with higher buying prices than selling prices, learning to both buy energy when the wholesale price is low as well as maximising the utilisation of the RES will be key to the agents' performance. Therefore, rewarding the agents based on their energy savings encourages both effective energy arbitrage and RES integration.

2.1.4. External microgrid trading

In a second case study, the microgrid aggregator (MGA) agent sells energy to five smaller xMGs. The MGA decides the amount of energy to sell at the next hour to the xMGs which then enter a bidding phase to compete for the energy.

The xMGs bid for a quantity of energy at a price they are willing to pay. The MGA then sells the energy to whichever xMGs bid the highest until there is none left to sell. The MGA can also set a reserve price which must be met to sell the energy, otherwise it is sent back to the utility grid.

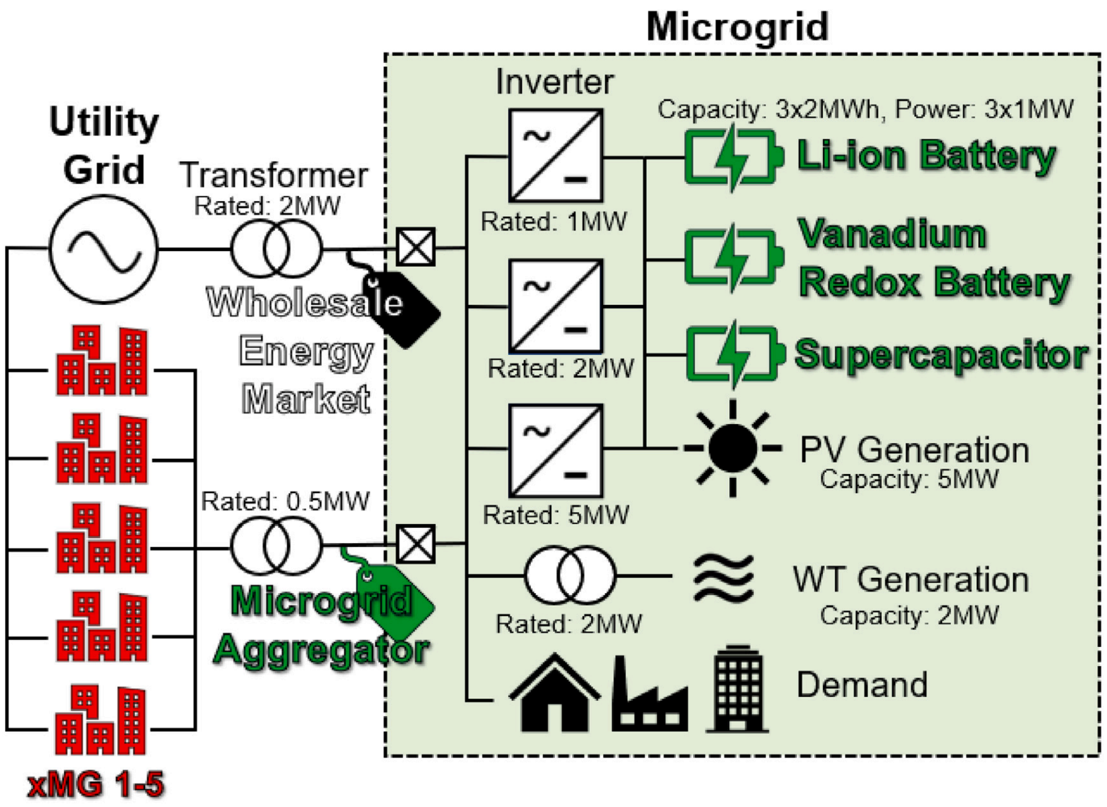


Fig. 1. Basic schematic of the microgrid.

2.2. *Control methodology*

Model predictive control had historically been the most widely used method for a control system, and is very popular within the energy industry [11]. Model-based methods such as this will often achieve great results if an accurate model and data are provided, but the results are only as accurate as the model itself and the data needed to create a model is not necessarily always available. In contrast, RL is a rapidly expanding research area and most methods are model-free so do not require a model of the environment, or any information about the environment at all, so are much more flexible in nature. Therefore, this paper focuses on the use of RL.

A fundamental challenge for this specific scenario is sample efficiency as there is only one set of data so the agents will only receive one pass through the environment. Therefore, RL algorithms known for their sample efficiency are required to achieve superior energy savings.

3. Reinforcement learning methodology

This section will explore the theory behind RL and introduce the specific algorithms used in the case studies later in the paper.

3.1. *Fundamentals*

Each time-step t in RL, the agent observes the current state of the environment s_t and selects an action a_t from an action-space following a learnt policy. The agent then receives a reward r_t from a reward function and transitions to a new state s_{t+1} with the goal of the agent being to learn a policy that maximises its total discounted future reward.

RL algorithms can be categorised as value function methods, policy gradient methods, or actor–critic methods. Value function methods evaluate performance by assigning value to each state by estimating the state-value $V(s)$ or state–action pair by estimating the action-value $Q(s,a)$. Policy gradient methods instead directly parameterise the policy using an ANN without necessarily estimating a value function. The parameters θ are adjusted to maximise an objective $J(\theta)$ by following the gradient of the policy $\nabla_\theta J(\theta)$. Actor–critic methods will both learn a value function and parameterise the policy, combining both approaches.

3.2. *Algorithms*

The algorithms used in this paper are Deep Deterministic Policy Gradient (DDPG) and two more advanced variants of DDPG, benchmarked against a popular value function method.

3.2.1. *Deep Deterministic Policy Gradient (DDPG)*

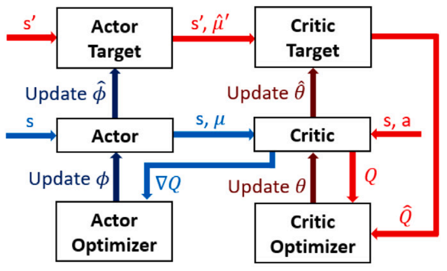
DDPG [27] is an actor–critic method based on the principles of Q-learning [28] in which an actor network $\mu_\phi(s)$ with network weights ϕ selects the actions the agent takes while a critic network $Q_\theta(s,a)$ with weights θ evaluates agent performance.

Action selection is performed by passing the current state through the actor network. As the policy is deterministic, the agent explores by adding a noise process w to the actor output:

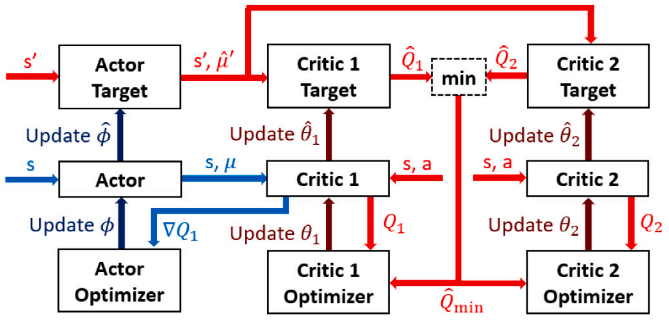
$$a_t = \mu_\phi(s_t) + w \tag{1}$$

This noise w is typically either Gaussian for uncorrelated noise or the Ornstein–Uhlenbeck process for correlated noise. An alternative method is to instead use adaptive parameter space noise which can be tuned during training [29]. NoisyNet layers [30] have been used with Deep Q-Networks (DQN) [31] to introduce noise to the value function estimation which the agent can learn to increase or decrease as it visits those states. The same approach is used in this work, and the noise process for the action selection has been removed.

Once the environment has executed the action, the agent stores the transition tuple $\langle s_t, a_t, r_t, s_{t+1}, d_t \rangle$ in an experience replay buffer memory, which can then be sampled later during training. This makes



(a) Learning process of DDPG.



(b) Learning process of TD3.

Fig. 2. Learning at each step when provided with a batch of transitions $\langle s_t, a_t, r_t, s_{t+1}\rangle$. The red arrows show the process for training the critic network and the blue arrows for training the actor network.

the algorithm an off-policy method as the agent is trained from samples collected using a different policy to the current learnt one. The use of a replay buffer allows DDPG to be more sample efficient than on-policy RL algorithms such as A3C [32] and PPO [33]. This is particularly useful for this environment as the agents only perform one sweep of the 4 years of data as one episode, so the ability to learn by sampling previous transitions from limited data is crucial.

Target actor $\hat{\mu}_{\hat{\phi}}$ and target critic $\hat{Q}_{\hat{\theta}}$ networks are also used to stabilise learning, with weights that are fixed during training and then updated at the end of each step. This is so that the agent is not calculating the target value using the same network that is being trained, which is done to avoid oscillations or divergence in the policy [31].

The critic and target critic networks evaluate performance by observing both the state and the corresponding action. Following the principles of temporal difference learning [8], target values y are estimated from the target critic estimations of the sampled batch of tuples:

$$y_i = r_i + \gamma \hat{Q}_{\hat{\theta}}\left(s_{i+1}, \hat{\mu}_{\hat{\phi}}(s_{i+1})\right) \tag{2}$$

The critic network is updated through gradient descent using the mean-squared error between y and the predicted value from the critic network:

$$\mathcal{L}_i(\theta) = \mathbb{E}\left[(y_i - Q_\theta(s_i, a_i))^2\right] \tag{3}$$

The actor network is then updated through gradient ascent using the deterministic policy gradient [34]:

$$\nabla_\phi J_i(\phi) = \mathbb{E}\left[\nabla_a Q_\theta(s_i, a)|_{a=\mu(s)} \nabla_\phi \mu_\phi(s_i)\right] \tag{4}$$

At the end of each step, soft target updates are performed for the target actor and critic using a smoothing parameter τ:

$$\begin{aligned}\hat{\theta} &\leftarrow \tau\theta + (1-\tau)\hat{\theta}\\ \hat{\phi} &\leftarrow \tau\phi + (1-\tau)\hat{\phi}\end{aligned} \tag{5}$$

The pseudocode of DDPG can be found in Algorithm 1. The learning process during training between the different ANNs for DDPG is visualised as a flowchart in Fig. 2(a).

3.2.2. Distributional DDPG (D3PG)

An issue with using the expectation value for $Q(s,a)$ is that it can over-generalise in environments with a non-deterministic reward function [35]. D3PG [36] removes the expectation step in the Bellman equation and instead estimates the value distribution $Z(s,a)$:

$$Z(s_t, a_t) \overset{D}{=} \mathcal{R}(s_t, a_t) + \gamma Z(s_{t+1}, a_{t+1}) \tag{6}$$

A number of atoms are assigned to each action for $N_{\text{actions}} \times N_{\text{atoms}}$ output neurons with the probability distribution of the return $d(s,a)$ calculated using a softmax applied separately across the atoms for each action. The algorithm C51 [35], the distributional variant of DQN, uses 51 atoms across each action. The probability mass on each atom is equal to:

$$d_i(s,a) = \frac{\exp(\theta_i(s,a))}{\sum_j \exp(\theta_j(s,a))} \tag{7}$$

Each of these probability masses can then be projected onto a support z with a value equally spaced between a minimum return $v_{\min}$ and a maximum return $v_{\max}$:

$$z_i = v_{\min} + (i-1)\frac{v_{\max} - v_{\min}}{N_{\text{atoms}} - 1} \tag{8}$$

The sum of these probability masses on each support is equal to $Q(s,a)$. The target used for training is the projected target distribution calculated using the categorical algorithm [35]:

$$(\Phi\hat{\mathcal{T}} Z_{\hat{\theta}}(s,a))_i = \sum_{j=0}^{N_{\text{atoms}}} \left[1 - \frac{|[\hat{\mathcal{T}} z_j]_{v_{\min}}^{v_{\max}} - z_i|}{\Delta z}\right]_0^1 \hat{d}_j(s_{i+1}, a^*) \tag{9}$$

where Φ is the projection of the distribution onto z and $\mathcal{T}$ is the distributional Bellman operator. As this generates a discrete probability distribution, the Kullback–Leibler divergence between the predicted value and the projected target is used for the loss function:

$$\mathcal{L}_i(\theta) = D_{\text{KL}}\left(\Phi\hat{\mathcal{T}} Z_{\hat{\theta}}(s_i, a_i) \parallel Z_\theta(s_i, a_i)\right) \tag{10}$$

3.2.3. Twin Delayed DDPG (TD3)

A common issue with using functional approximators in RL is they have a tendency to overestimate value estimates which can lead to

Algorithm 1 Deep Deterministic Policy Gradient

```
1: Initialise critic $Q_\theta(s,a)$ and actor $\mu_\phi(s)$ arbitrarily
2: Initialise target critic $\hat{Q}_{\hat{\theta}}(s,a)$, setting $\hat{\theta} \leftarrow \theta$
3: Initialise target actor $\hat{\mu}_{\hat{\phi}}(s)$, setting $\hat{\phi} \leftarrow \phi$
4: Initialise replay memory
5: for $t = 0, T$ do
6:     Observe state $s_t$
7:     Choose action $a_t = \mu_\phi(s_t)$
8:     Execute $a_t$, observe reward $r_t$ and next state $s_{t+1}$
9:     Store transition $\langle s_t, a_t, r_t, s_{t+1}\rangle$ in memory
10:    Sample minibatch $\langle s_i, a_i, r_i, s_{i+1}\rangle$ from memory
11:    Calculate target $y_i = r_i + \gamma\hat{Q}_{\hat{\theta}}(s_{i+1}, \hat{\mu}_{\hat{\phi}}(s_{i+1}))$
12:    Critic loss $\mathcal{L}_i(\theta) = \mathbb{E}\left[(y_i - Q_\theta(s_i, a_i))^2\right]$
13:    Perform gradient descent on $\theta$ using $\mathcal{L}(\theta)$
14:    Actor loss $\nabla_\phi J(\phi) = \mathbb{E}\left[\nabla_a Q_\theta(s,a)|_{a=\mu(s)} \nabla_\phi \mu_\phi(s)\right]$
15:    Perform gradient ascent on $\phi$ using $\nabla_\phi J(\phi)$
16:    Update critic weights $\hat{\theta} \leftarrow \tau\theta + (1-\tau)\hat{\theta}$
17:    Update actor weights $\hat{\phi} \leftarrow \tau\phi + (1-\tau)\hat{\phi}$
18: end for
```

suboptimal policies as agents tend to select the actions that they have already assumed are more valuable [37]. Double DQN [38] looked to solve this by removing the maximisation step in the DQN target calculation and selecting the next state's best action using the evaluation network, but this does not work for actor–critic methods such as DDPG as the action selection and value function estimates are separate. Instead, Double Q-Learning [39] uses two value function estimates so that the next best action and target estimate is performed using different value estimates.

TD3 [40] aims to solve the overestimation problem for DDPG by using a similar approach with a second critic and target critic pair. The target value y_i is then calculated using the minimum estimated of the two critic target networks:

$$y_i = r_i + \gamma \min_{i=1,2} \hat{Q}_{\hat{\theta}_i}(s_{i+1}, \hat{\mu}_{\hat{\phi}}(s_{i+1})) \tag{11}$$

That shared target value y is then used for the loss function of both critic networks:

$$\begin{aligned} \mathcal{L}_i(\theta_1) &= \mathbb{E}\left[(y_i - Q_{\theta_1}(s_i, a_i))^2\right] \\ \mathcal{L}_i(\theta_2) &= \mathbb{E}\left[(y_i - Q_{\theta_2}(s_i, a_i))^2\right] \end{aligned} \tag{12}$$

In contrast to DDPG where both the actor and critic are updated every step, the actor and target networks are updated at a lower frequency to the critic to minimise the error in the policy update. The actor update uses the deterministic policy gradient of only one of critics:

$$\nabla_\phi J_i(\phi) = \mathbb{E}\left[\nabla_a Q_{\theta_1}(s_i, a)|_{a=\mu(s)} \nabla_\phi \mu_\phi(s_i)\right] \tag{13}$$

The learning process of TD3 is visualised as a flowchart in Fig. 2(b).

3.3. Multi-Agent DDPG (MADDPG)

In theory, using a single-agent RL for a MAS is possible as the state observations, actions, and reward function can all remain the same. However, there are a number of fundamental challenges that appear when applying standard single-agent RL algorithms in a MAS.

The main problem is that the environment appears non-stationary from the perspective of a single agent as the behaviours of other agents will change during training [41]. This means that an agent may observe exactly the same state as before and perform exactly the same action but receive a completely different reward based on the actions of other agents, which can make learning incredibly difficult. For example, agents in a competitive setting may over-fit their policy based on the behaviours of other agents, whereas agents in a collaborative setting may suffer from credit assignment problems from a shared reward function.

MADDPG [42] looks to solve this by using the principle of centralised learning with decentralised execution. Each agent selects its own action based on its own observations during the execution phase, but the architecture of the DDPG critic allows each agent to evaluate performance by observing the actions taken by all other agents, thus solving the problem where agents may perceive the environment as non-stationary.

As each of the agents possesses its own critic network, they can evaluate their own performance using their own reward function. This means that MADDPG can be used in competitive, collaborative, and mixed settings while remaining fundamentally very similar to DDPG itself.

MADDPG has the benefit of other multi-agent RL algorithms in that it uses a critic network for each agent, rather than having a shared critic as in QMIX [43] or COMA [44] which are only suited for a cooperative MAS. MADDPG allows each agent to learn its own reward function so is much better suited for competitive or mixed cooperative and cooperative environments.

3.4. Environment formulation

For RL to operate the microgrid environment, the control process must be generalised to a state-space $\mathcal{S}$, action-space $\mathcal{A}$, reward function $\mathcal{R}$, and a state transition function. The rest of this section will explain the states, actions, and rewards for the case studies.

3.4.1. States

The state observations s_t are the values the agents see when selecting actions. The ESS agents observe the charge of all of the ESSs, the grid demand, wholesale energy price, WT and PV generation, as well as the current hour in the day and the week. The agents also receive predicted values of the next step for demand $\dot{\mathrm{X}}^{\mathrm{D}}_{t+1}$, market price $\dot{\mathrm{P}}^{\mathrm{grid}}_{t+1}$, and PV $\dot{\mathrm{X}}^{\mathrm{PV}}_{t+1}$, and WT $\dot{\mathrm{X}}^{\mathrm{WT}}_{t+1}$ from ANNs using regression with weather data, as in previous work [45]. In the second case study, the MGA agent receives the same observations as the ESS agents but also includes observations of the MGA sell volume and reserve price for that step:

$$\begin{aligned} s^{\mathrm{ESS}}_t = [&c^{\mathrm{LIB}}_t, c^{\mathrm{SC}}_t, c^{\mathrm{VRB}}_t, \mathrm{P}^{\mathrm{grid}}_t, \dot{\mathrm{P}}^{\mathrm{grid}}_{t+1}, \mathrm{P}^{\mathrm{MGA}}_t, \mathrm{X}^{\mathrm{MGA}}_t, \\ &\mathrm{X}^{\mathrm{D}}_t, \dot{\mathrm{X}}^{\mathrm{D}}_{t+1}, \mathrm{X}^{\mathrm{PV}}_t, \dot{\mathrm{X}}^{\mathrm{PV}}_{t+1}, \mathrm{X}^{\mathrm{WT}}_t, \dot{\mathrm{X}}^{\mathrm{WT}}_{t+1}, \mathrm{H}^{\mathrm{day}}_t, \mathrm{H}^{\mathrm{week}}_t] \end{aligned} \tag{14}$$

In contrast, the xMG agents only receive the demand for that specific xMG, the MGA sell volume and reserve price, as well as the same hour information as before:

$$s^{\mathrm{xMG}}_t = [\mathrm{P}^{\mathrm{MGA}}_t, \mathrm{X}^{\mathrm{MGA}}_t, \mathrm{X}^{\mathrm{xMG,D}}_t, \mathrm{H}^{\mathrm{day}}_t, \mathrm{H}^{\mathrm{week}}_t] \tag{15}$$

This means that aspects of the underlying state of the environment are hidden from different agents, in the interests of privacy and competition. All observation values are also scaled between 0 and 1 for stable agent training.

3.4.2. Actions

The actions of the agents control the environment at each step. DDPG and its variants output an action a value between −1 and 1, and then that value is used to control the environment.

The ESS agents control how much that ESS charges or discharges. If using a single-agent approach, one agent will output three actions (or five including the MGA) to charge all three ESSs, whereas each agent will be in control of one ESS in a multi-agent approach:

$$x^{\mathrm{ESS}}_t = \mathrm{X}^{\mathrm{ESS}}_{\max} \left[a^{\mathrm{ESS}}_t\right]^{1}_{-1} \tag{16}$$

The MGA agent selects two actions. The first selects the primary microgrid's selling volume at the next step, between 0 and a maximum volume $\mathrm{X}^{\mathrm{MGA}}_{\max}$:

$$\mathrm{X}^{\mathrm{MGA}}_{t+1} = 0.5\mathrm{X}^{\mathrm{MGA}}_{\max} \left[a^{\mathrm{MGA,X}}_t\right]^{1}_{-1} + 0.5\mathrm{X}^{\mathrm{MGA}}_{\max} \tag{17}$$

The second decides the reserve price of the energy sold, between $\mathrm{P}_{\min} = £16/\mathrm{MWh}$ and $\mathrm{P}_{\max} = £144/\mathrm{MWh}$:

$$\begin{aligned} \mathrm{P}^{\mathrm{MGA}}_{t+1} = {} & 0.5\left(\mathrm{P}_{\max} - \mathrm{P}_{\min}\right) \left[a^{\mathrm{MGA,P}}_t\right]^{1}_{-1} \\ & + 0.5\left(\mathrm{P}_{\max} + \mathrm{P}_{\min}\right) \end{aligned} \tag{18}$$

The xMGs also select two actions. The first is the bid volume, between 0 and a maximum volume $\mathrm{X}^{\mathrm{xMG}}_{\max}$:

$$x^{\mathrm{xMG}}_t = 0.5\mathrm{X}^{\mathrm{xMG}}_{\max} \left[a^{\mathrm{xMG,X}}_t\right]^{1}_{-1} + 0.5\mathrm{X}^{\mathrm{xMG}}_{\max} \tag{19}$$

The second is the bid price, also between $\mathrm{P}_{\min}$ and $\mathrm{P}_{\max}$:

$$\begin{aligned} p^{\mathrm{xMG}}_t = {} & 0.5\left(\mathrm{P}_{\max} - \mathrm{P}_{\min}\right) \left[a^{\mathrm{xMG,P}}_t\right]^{1}_{-1} \\ & + 0.5(\mathrm{P}_{\max} + \mathrm{P}_{\min}) \end{aligned} \tag{20}$$

In the second case study, the MGA decides the sell volume of energy and the reserve price for the xMG at the next step. These values are then observed by all agents at the next step. In this case, the global controller will output five actions instead of three, whereas the individual MGA agent will operate as its own entity outputting two actions in the multi-agent setting. Each xMG agents decide two actions: the buying volume of energy and the bid price of the current step.

For DDPG, the primary algorithm used in this paper, the actions are selected using an actor network for each agent which observes only the relevant state observations for that particular agent. However, the agents' performance is evaluated using a critic network for each agent which receives those same observations as well as the actions taken by all agents.

3.4.3. Rewards

The aim of the agents is to reduce the energy costs of their respective microgrid. For the primary microgrid, this is done by calculating the energy imported to or exported from the grid. First, the net demand of the DC line X^{dc} is considered:

$$X_t^{dc} = x_t^{LIB}\eta_{RTE}^{LIB} + x_t^{VRB}\eta_{RTE}^{VRB} + x_t^{SC}\eta_{RTE}^{SC} - X_t^{PV} \tag{21}$$

This is then used to calculate the amount of energy imported from or exported to the utility grid:

$$X_t^{in} = (X_t^{D} + X_t^{dc}\eta_{[1,2,5]MW}^{inv} - X_t^{WT}\eta_{2MW}^{tra})\eta_{2MW}^{tra} \tag{22}$$

The reward for the cost of the energy the primary microgrid has bought or sold to the utility grid can then be calculated:

$$R_t^{in} = -X_t^{in}P_t^{grid} \tag{23}$$

The ESS agents are punished to promote positive behaviour, such as by including the ESS operation cost R^{CPC} calculated earlier in Eq. (29). The agent is punished again if the action selected would make the theoretical ESS charge $\dot{c}$ after the action is executed exceed the capacity boundaries of the ESS:

$$R_t^{cap} = \begin{cases} P_{max}(\dot{c}_t)^2/X_{max} & \dot{c}_t < 0 \\ 0 & 0 \le \dot{c}_t \le C_{max} \\ P_{max}(\dot{c}_t - C_{max})^2/X_{max} & \dot{c}_t > C_{max} \end{cases} \tag{24}$$

The agents are also punished based on the amount of energy lost each step through self-discharge:

$$R_t^{SDC} = P_{max}\left(\frac{c_t}{C_{max}}\right)\eta_{SDC} \tag{25}$$

Including the ESS operation cost R^{CPC} from Eq. (29) and the reward for the MGA bidding R^{MGA} from Algorithm 2, the sum of the reward terms R^{sum} is calculated as:

$$R^{sum} = R^{in} + R^{MGA} - R^{CPC} - R^{SDC} - R^{cap} - R^{base} \tag{26}$$

This value is then multiplied by 0.01 and the total number of agents N_{agents} so that the reward the agent receives typically remains between −1 and 1, or at least at that magnitude, to improve the stability of agent learning. Therefore, the reward r_t an agent receives at each step is equal to:

$$r_t = 0.01N_{agents}R^{sum} \tag{27}$$

Using the raw energy cost as the reward will punish the agent when the demand is high and reward the agent when RES generation is high, neither of which the agent has any control over. Therefore, a baseline R^{base} is used to normalise the reward for effective learning. The single-agent algorithms have control over the whole environment so are rewarded on the primary microgrid's energy savings with a R^{base} of if all of the ESS and MGA had remained idle by recalculating what the reward would be if x^{LIB}, x^{LIB}, x^{LIB}, and X^{MGA} were all 0.

However, the multi-agent methods can be more flexible with reward function design. For example, each agent could be rewarded on the entire primary microgrid's savings as this is the common goal the agents are working towards but this could lead to credit assignment problems as the agents will need to assess how much of the shared reward they contributed to. Alternatively, they could be rewarded based on their own individual savings, but that then ignores the policies of the other ESSs and may lead to a lower global reward.

Therefore, both approaches can be combined by rewarding the agent on the grid's savings but by using a baseline of if that one individual agent had remained idle, rather than if all of them together were idle. This follows the principle of marginal contribution in cooperative game theory where agents are rewarded based on their contribution to the global goal [46].

4. Case studies

In the first case study, only the ESS agents are used and their actions control how much each ESS charges or discharges by. In the second, the MGA agent determines the selling volume and reserve price for the next step which all agents observe before the xMG agents place their bids for the amount and price.

4.1. Environment parameters

This section will cover the technical characteristics and dynamics of the RL environment.

4.1.1. ESS properties

The parameters used for the modelling of each ESS type are given in Table 1 with their maximum capacity C_{max}, maximum power X_{max}, self-discharge efficiency (SDC) per hour η_{SDC}, round-trip efficiency (RTE) η_{RTE}, capacity cost per kWh, number of lifecycles, and capacity cost per cycle P_{CPC}.

The charge of each ESS is bound between 0 and their maximum capacity C_{max} = 2MWh, with the power x_t the ESS can charge or discharge each step bound between 0 and their maximum power X_{max} = 1 MW. The charge of each ESS c_t each step is calculated by:

$$c_t = x_t\sqrt{\eta_{RTE}} + c_{t-1}\eta_{SDC} \tag{28}$$

The operation CPC of each ESS is also considered so that the agents optimally use the different types. This is calculated using their capacity, capacity cost, and number of lifecycles which is then used as a function in the reward R^{CPC}:

$$R_t^{CPC} = 0.5P_{CPC}\left(\frac{c_t - c_{t-1}}{C_{max}}\right)^2 \tag{29}$$

A cycle is considered as the battery charging from empty to full and then discharging until empty, or vice versa. Therefore, P_{CPC} is halved as each step is only considered as a half cycle because the ESS cannot both charge and discharge in the same step.

4.1.2. RES output

The output for the RES generation is modelled using weather data collected at the Keele University weather station. The PV output X^{PV} is calculated using hourly solar radiation data and scaled to model a solar farm with a maximum capacity of 5MW, while the WT output X^{WT} is calculated using wind power curve modelling using wind speed data, with wind speed v:

$$X^{WT} = \begin{cases} 0 & v < v_{ci} \text{ or } v > v_{co} \\ \frac{1}{2}\rho\pi r^2 c_p v^3 & v_{ci} \leqslant v < v_r \\ X_{rated}^{WT} & v_r \leqslant v \leqslant v_{co} \end{cases} \tag{30}$$

The values used are v_{ci} = 3 ms^{-1}, v_r = 12 ms^{-1}, and v_{co} = 25 ms^{-1} are the cut-in, rated, and cut-out wind speeds respectively which are comparable to a 1MW-rated turbine with a blade radius of r = 30 m and a power coefficient c_p = 0.4 [47]. This microgrid considers two of these turbines for a combined maximum wind capacity of 2 MW. The same power curve approach is used by both Kuznetsova et al. [13] and Zhang et al. [16] for their RES output.

Table 1
ESS Properties.

ESS	Capacity C_{max}	Power X_{max}	η_{SDC}	η_{RTE}	Capacity Cost	Lifecycles	P_{CPC}
LIB	2MWh	1MW	99.99%	95%	£100/kWh	5k	£40
VRB	2MWh	1MW	100%	80%	£200/kWh	10k	£40
SC	2MWh	1MW	99%	95%	£300/kWh	100k	£6

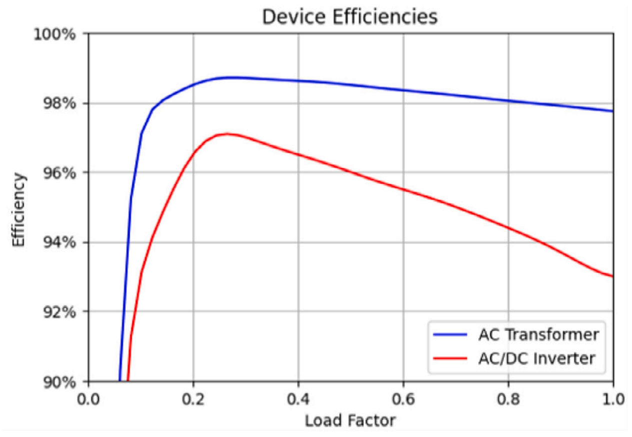


Fig. 3. Inverter and transformer efficiency profiles.

4.1.3. *Transformer and inverter properties*

The primary microgrid is a hybrid AC/DC network with inverters between the two power lines and transformers to step-up and step-down the voltage to the microgrid. The rated loads of each of the transformers and inverters can be found in Fig. 1.

The HESS and PV generation are connected to 1MW, 2MW, and 5MW rated inverters as well as to each other. When energy is to pass between the AC and DC lines, the grid will automatically use whichever inverter or combination of inverters that would result in the lowest power loss. Mbuwir et al. [21] used a similar efficiency profile for inverters [48] for RL in a microgrid with solar energy.

The efficiency profiles modelled using a polynomial estimation, shown in Fig. 3. The efficiency profiles are nonlinear and given as a function of the load factor, which is given as the current load divided by the rated load. Although the real efficiency of these devices at low load factor would approach 0%, the minimum efficiency is capped at 10% to prevent divisions by 0 during environment calculations.

4.1.4. *Wholesale energy market*

The microgrid operates under a dynamic energy pricing scheme where the prices P^{grid} are set by a real-time energy trading market covering the UK [49]. However, there is also a set maximum price P_{max} = £144/MWh as this is the average unit rate for electricity for the UK [50] so will instead buy from the utility grid at that flat price if the market price exceeds this.

However, this market is only for purchasing energy so the microgrid can also sell energy back to the utility grid at a fixed feed-in tariff. The feed-in tariff in the UK varies greatly based on the type of energy, the size of the system, and the installation date [51]. Therefore, a fixed rate of P_{min} = £16/MWh is assumed for all energy sold back to the grid. This is equal to the tariff for a WT system selling between 0.1 MW and 1.5 MW installed after 1st January 2019, and is also extremely similar to the £15.9/MWh feed-in tariff of a 0.25 MW and 1 MW PV system installed at the same time [52].

4.2. *MGA properties*

The MGA selects the total volume the grid will sell X^{MGA} for the xMGs to bid for. The MGA also selects a reserve price P^{MGA} between P_{min} and P_{max} which needs to be met in order to sell so that the MGA has some control over the xMG market.

During the bidding phase of each step, the aggregator will sell as much available energy as available to the highest xMG bidder up to the volume it has requested, assuming the xMG bid price exceeds the MGA reserve price. The MGA will then move to the next highest bidder until all available energy for selling has gone, with any excess energy put up for auction but not sold given back to the utility grid at P_{min}. This is written in pseudocode in Algorithm 2.

Algorithm 2 MGA and xMG Bidding Step

```
1: Input MGA selling volume $X^{MGA}$
2: Input MGA reserve price $P^{MGA}$
3: Input xMG bidding volumes
     $x^{xMG} = [x^{xMG1}, x^{xMG2}, x^{xMG3}, x^{xMG4}, x^{xMG5}]$
4: Input xMG bidding prices
     $p^{xMG} = [p^{xMG1}, p^{xMG2}, p^{xMG3}, p^{xMG4}, p^{xMG5}]$
5: Initialise reward $R^{MGA} = 0$
6: while $X^{MGA} > 0$ and $x^{xMG}_{any} > 0$ do
7:    Highest bidding xMG index $i = \text{argmax}\, p^{xMG}$
8:    if $p_i^{xMG} > P^{MGA}$ then
9:       Available bid volume $x^{bid} = \min[X^{MGA}, x_i^{xMG}]$
10:      Update reward $R^{MGA} \leftarrow R^{MGA} + 0.8 p_i^{xMG} x^{bid}$
11:      Update selling volume $X^{MGA} \leftarrow X^{MGA} - x^{bid}$
12:   end if
13:   Set $x_i^{xMG} \leftarrow 0$
14:   Set $p_i^{xMG} \leftarrow 0$
15: end while
16: Sell remainder $R_{MGA} \leftarrow R_{MGA} + P_{min} X^{MGA}$
```

4.2.1. *xMG properties*

Each xMG must satisfy its own demand $X^{xMG,\,D}$ and can do so by trading with the MGA or by importing from the utility grid. The demand is one-twentieth of the scale of the primary microgrid's demand $X^{xMG,D}$ plus a small noise function:

$$X_t^{xMG,\,D} = \left[0.05X_t^D + \mathcal{N}(0, 0.01)\right]_{0.01X_t^D}^{0.25X_t^D} \tag{31}$$

The xMG bids for a volume of energy x^{xMG} at a price p^{xMG} between P_{min} and P_{max}. Any demand that is not met by the volume it receives from the MGA is made up by buying from the utility grid at P_{max}. Any energy bought over the xMG demand is given back to the utility grid at P_{min}.

4.3. *Simulation setup*

Each time-step in the environment represents one real-world hour and each episode represents 168 h or one week. The entire simulation takes place over 200 weeks for 33.6k total time-steps, significantly fewer samples than used for typical RL agent training in the order of millions of time-steps. Therefore, sample efficiency is key to agent performance.

The agent begins learning after 500 steps to build up the replay buffer and takes uniformly random actions until 1000 steps so that the initial transitions are not biased by the randomly initialised weights of the actor network. The agent then trains across all 200 weeks with the

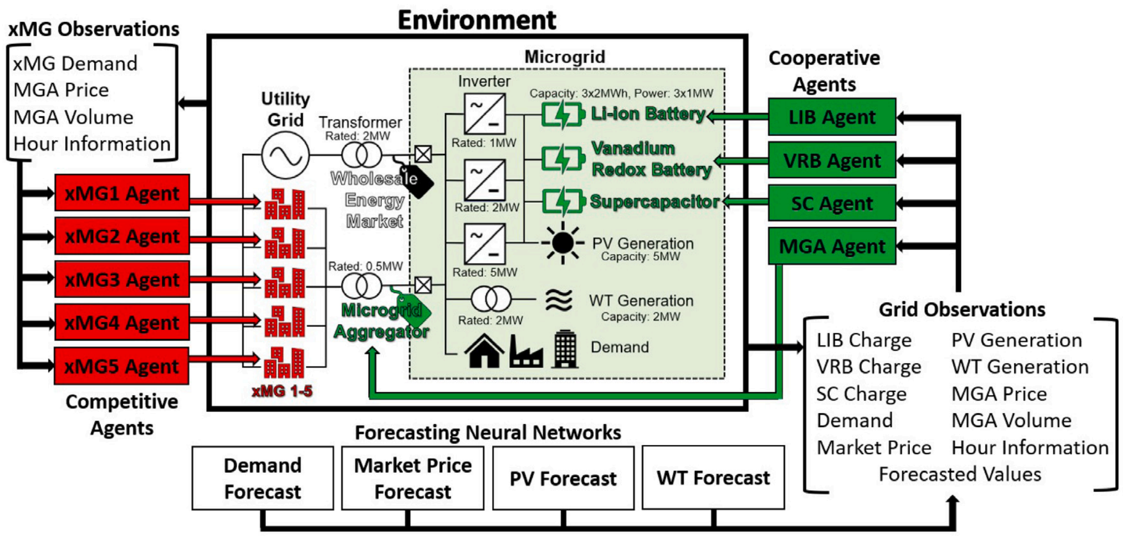


Fig. 4. Illustration of how each agent interacts with the environment in the xMG trading case study.

Table 2
DDPG agent hyperparameters.

Hyperparameter	Value
Actor Learning Rate	5×10^{-4}
Critic Learning Rate	1×10^{-3}
Hidden Layer Dimensions	256
Hidden Layer Activation	ReLu
Batch Size	64
Target Update τ	0.005
Discount Factor γ	0.99
D3PG Limits $V_{\min}$, $V_{\max}$	−10, 10
TD3 Actor Delay Steps	2

first 100 used for hyperparameter testing and the second 100 weeks for evaluation.

The hyperparameter testing was performed using a grid search of values for the DDPG algorithm with. The parameters tested were the actor and critic learning rates, the discount factor γ, the hidden layer dimensions, the learning sample batch size, and the target network update τ. During this testing, the grid savings are reset after 1000 steps due to the random action selection and then the value that produced the greatest savings after the 100 episodes is carried forward for the grid search of the next hyperparameter. The final values used for evaluation shown in Table 2 with those parameter values used across all DDPG variants.

For the evaluation phase, the agent is trained across all 200 episodes with the savings reset after the first 100. The *TensorFlow* and *NumPy* random seeds are kept constant across all simulations for fairness and repeatability, so every agent takes the same random actions in the first 1000 steps.

Forecasted values are predicted using regression ANNs for the demand, price, and RES generation. In previous work [45], this was found to significantly boost agent performance in a similar environment performing energy arbitrage. These forecasts are then provided to the agents as state observations.

At each time step, the environment and forecast ANNs provide the agents with observations, and then they all select an action at the same time. The environment executes all actions at once, then providing the agents with their rewards and transitioning to the next state. The control process at each step is found in Fig. 4.

4.4. Benchmarks

Three benchmarks are used for evaluation in the HESS case study: two alternative multi-agent RL algorithms and a model-based approach. Deterministic optimisation methods such as linear programming cannot be used in this environment as the energy price is dependant on whether the agent is buying or selling, as well as the demand and RES. This makes the problem nonlinear and there are no effective methods of solving nonlinear programming problems [53].

No model-based benchmark methods are used for the xMG case study due to the competition between the MGA and the xMG agents. The two alternative multi-agent RL algorithms also can only use discrete action-spaces which give the continuous control of DDPG a tremendous advantage in this particular bidding setting. Therefore, the DDPG algorithms will be benchmarked against each other in the second case study.

4.4.1. Multi-Agent Deep Q-Networks (MADQN)

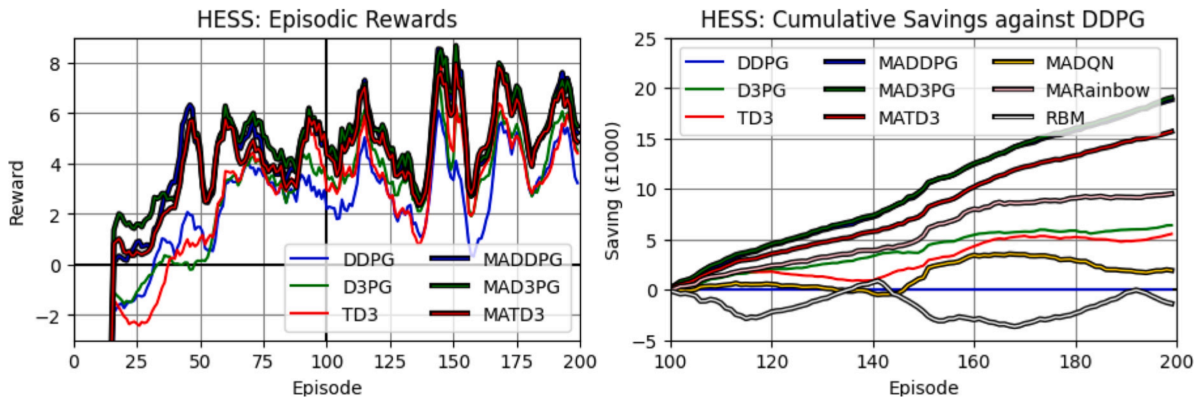
DQN [31] uses an ANN to evaluate the action values of different actions, similar to the critic network in DDPG. However, rather than using an actor network for action selection, the agent will always the select the most valuable action at each step unless told otherwise. This makes DQN more robust than actor–critic methods such as DDPG as the algorithms are less sensitive to hyperparameter tuning [8], but the agent can only select from a discrete action-space. Also, as the action selection and evaluation is performed using the same ANN, it is not as easy to apply DQN effectively as DDPG in a MAS.

The approach used for the benchmarks was developed by Tampuu et al. [54] in which each agent acts independently and learns its own action-value estimate. Therefore, the agents effectively consider all other agents as part of the environment. These agents receive the same reward as the DDPG and MADDPG agents based on each agents marginal contribution towards the shared objective of reducing the energy bills.

Both MADQN and multi-agent Rainbow (MARainbow) will be used as benchmarks. In previous work [45], Rainbow was found to be able to use a larger discrete action-space than regular DQN for ESS control where its greater learning properties and sample efficiency outperformed the continuous control of DDPG. Therefore, each MADQN agent selects from 5 actions whereas the MARainbow agents can select from 9 where the actions range from maximum charge of the ESS to maximum discharge.

4.4.2. Rule-based model (RBM)

The same rule-based approach was used as in previous work [55] in which the agent will always look to maximise the utilisation of the RES. This is standard practice for most ESSs looking to maximise RES,



(a) Episodic rewards, averaged over 5 episodes.

(b) Cumulative savings over each evaluation episode.

Fig. 5. Rewards and savings of Case Study 1.

Table 3
Results of Case Study 1, with the best result of each column highlighted.

Algorithm	Case Study 1			
	Sav.(£1k)	Adj.(£1k)	vs DDPG	ESS Loss
DDPG	42.92	34.39	–	19.87%
D3PG	48.78	40.78	18.58%	16.40%
TD3	46.69	39.93	16.11%	14.48%
MADDPG	**<u>59.58</u>**	53.24	54.81%	10.64%
MAD3PG	59.50	**<u>53.44</u>**	**<u>55.39%</u>**	**<u>10.18%</u>**
MATD3	56.19	50.06	45.57%	10.91%
MADQN	47.28	36.32	5.61%	23.18%
MARainbow	52.01	43.90	27.65%	15.59%
RBM	40.93	33.02	−3.98%	19.33

Table 4
Adjusted savings of the different approaches.

Algorithm	Case Study 1		
	SGC	MAS-G	MAS-MC
DDPG	34.39	40.32	53.24
D3PG	40.78	35.27	**<u>53.44</u>**
TD3	39.93	41.83	50.06

but ignores the dynamic energy prices of the wholesale market. The ESSs charge when RES generation exceeds demand and discharge when reversed. As the different ESSs are suited for different timescales of energy storage, the model charges and discharges the SC first, followed by the LIB, followed by the VRB.

5. Results and discussion

This section will present the results for the two case studies. Discussions will focus on the performance of the agents as well as different behaviours between algorithms, while assessing if the single-agent or multi-agent approaches were better.

The performance of the algorithms is divided into the raw savings, the savings adjusted for the cost of the ESS operation, percentage difference against DDPG, and the percentage of savings lost from the ESS. The second case study also considers the savings made directly from the MGA.

5.1. Case study 1: HESS

This study analysed the use of RL and multi-agent RL for the control of a HESS. The results for this case study can be found in Table 3, with the smoothed episodic rewards in Fig. 5(a) and adjusted savings over time plotted in Fig. 5(b). This case study showed that the agents are able to make a significant energy cost savings over two years, even with a low fixed selling tariff by maximising the value of the RES generated.

5.1.1. Algorithm performance

The multi-agent approaches using the improved game theory reward performed overwhelmingly better than the single-agent controllers and the multi-agent methods with shared rewards, with MAD3PG performing 55.4% better than DDPG. Not only were the multi-agent methods able to use the different ESSs effectively for a lower ESS loss, but were also able to make significantly higher savings before ESS operating cost is considered too. The individual reward functions allow the ESSs operating separately but collaboratively to be much more efficient than having all of them controlled together by a single controller.

However, the multi-agent approaches were only better by such an extent when each agent used the marginal contribution reward. Table 4 shows the adjusted savings for the different algorithms in both the single global controller (SGC) and MAS, including the results for if the MAS agents received the same global reward as the SGC or the individual MAS rewards using the marginal contribution baseline; MAS-G and MAS-MC respectively. All algorithms perform better using the MAS-MC but the effect from SGC to MAS-G is not nearly as consistent, including a notable drop in performance for D3PG. This means the benefit in using a MAS for this environment comes from being able to utilise better reward function design, rather than simply from the distributed control.

MADQN performs fairly well but is clearly limited by the small discrete action-space. Despite this, this method shows the value of using a multi-agent approach as it is still able to outperform DDPG and TD3 with the limitation. MARainbow performs better with the larger action-space but still cannot reach the level of the multi-agent DDPG-based methods as the reinforced inter-agent learning approach appears to be inferior to centralised learning.

RBM performs well but is completely passive on a number of weeks where RES never exceeds demand. This shows the RL agents are able to more effectively utilise the RES, but can also able to respond to the price signals by charging the ESS types when prices are lower in the morning to use later when the prices rise.

5.1.2. Agent behaviour

Most of the agents generally behave in a similar way. The ESSs will only really be used at times when there is a large amount of RES to store with the agents learning that any PV generation at its peak around noon is best to store so it can be used later when prices are higher. There is also some charging at the start of the day when prices are

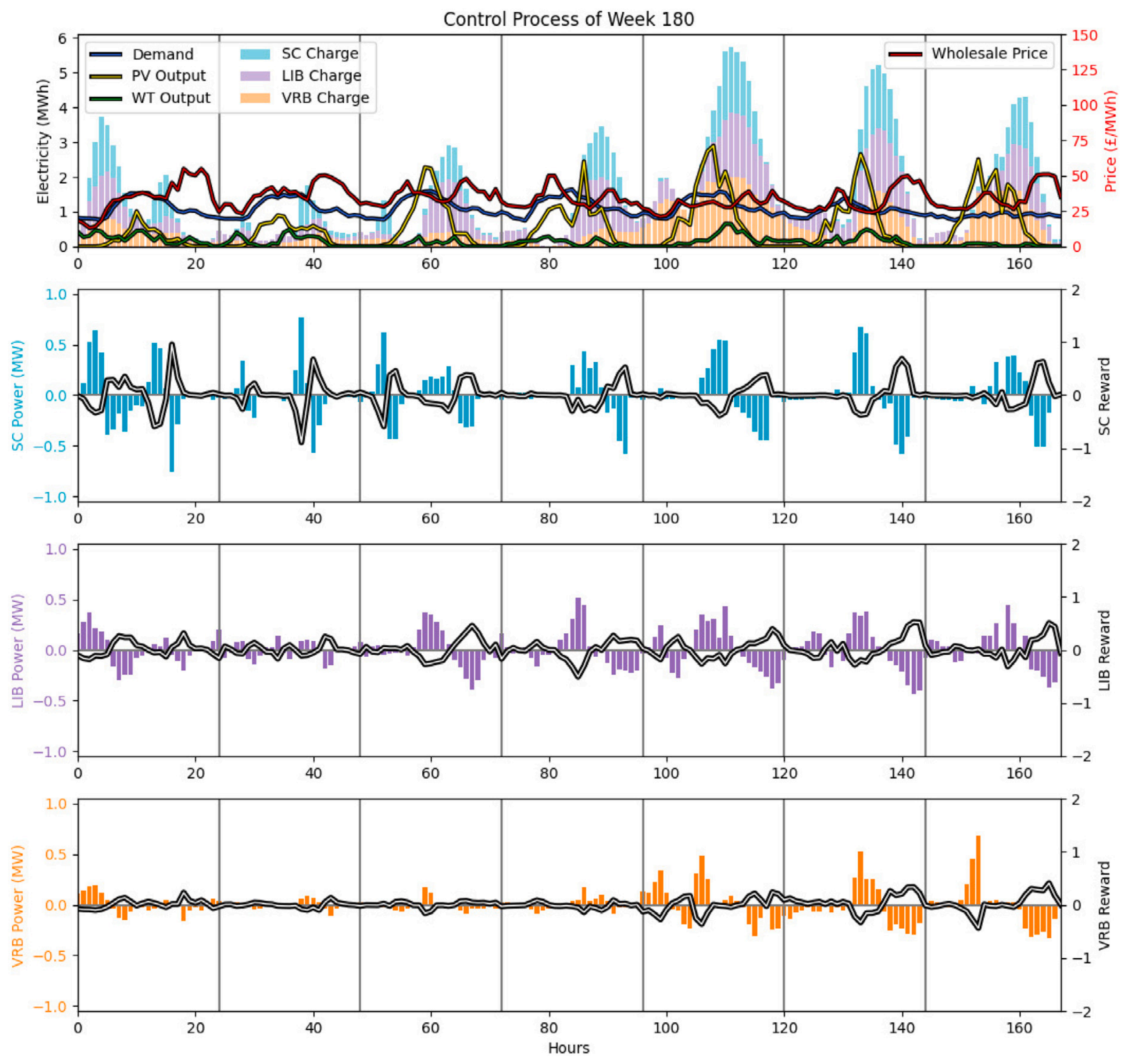


Fig. 6. Microgrid control in Episode 180 using MAD3PG.

lower, but this is less significant. Due to the battery cycling costs, agents very rarely charge or discharge at maximum power but instead look to charge and discharge over longer periods of time.

Almost every agent uses the LIB the most as its well-rounded properties make it ideal for storing over short, medium, and long-term. The D3PG, MADDPG, MATD3, MADQN, and MARainbow agents then use the SC the second most whereas for DDPG and TD3 it is the VRB. A clear difference between the single controller versus multiple agent methods is that the multi-agent algorithms showed a clear change in the behaviour for the different ESS types, whereas it was much less pronounced in the single agent algorithms.

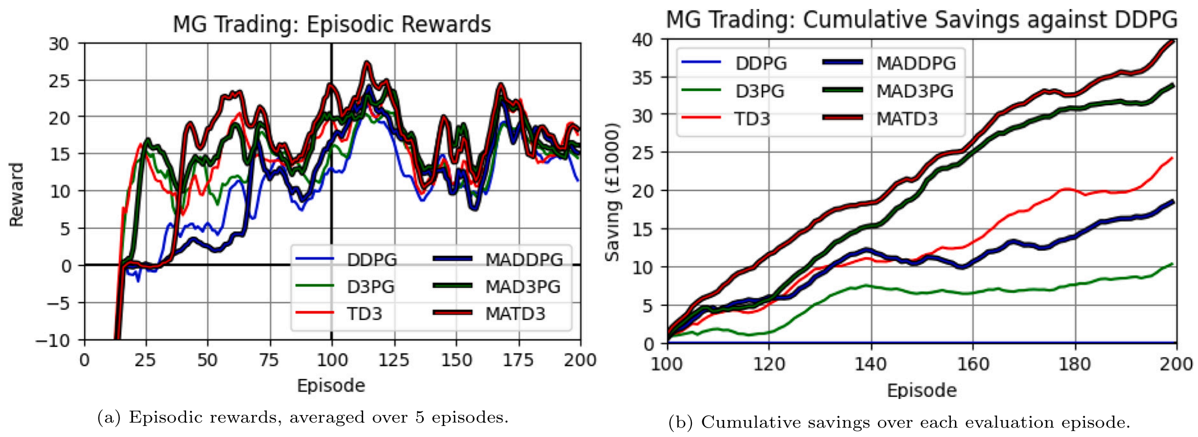
There are a couple of cases of unique behaviour between algorithms though. Most notably, the MAD3PG agent uses the SC the most aggressively, followed by the LIB and then the VRB which is interesting as its adjusted savings are the best of any other algorithm. The control process of the MAD3PG agent for week 180 is shown in Fig. 6, selected as the episode is towards the end of the simulation but has higher RES generation than later episodes.

Other examples of notable behaviours include D3PG which would look to store energy over greater lengths of time. However, as the demand and price peak cycle with a peak in the evening every day, it is more efficient to use as much stored energy as possible at the peak price times and end the day with minimal charge. Another example is the MATD3 VRB agent which stays almost completely idle all of the time. The RBM behaves as it is instructed, but the resulting issue is that it is completely inactive if RES does not exceed demand whereas the other agents still learn to charge when there are low prices in the morning regardless of RES.

The multi-agent algorithms performed better as the agents could learn a policy for each type of ESS, rather than a single policy for controlling every ESS. Generally, the single agent algorithms would operate each ESS as if they had the same properties because it is more difficult to differentiate the reward punishments for the multiple ESSs when there is only a single reward function. Therefore, the multi-agent approach should be considered superior as it allows the learning of better individual policies for each ESS.

5.2. Case study 2: xMG trading

The results for this case study can be found in Table 5, with the smoothed episodic rewards in Fig. 7(a) and adjusted savings over time plotted in Fig. 7(b). When compared to the results of Case Study 1, the tables shows very clearly that being able to sell energy on the grid's



(a) Episodic rewards, averaged over 5 episodes.

(b) Cumulative savings over each evaluation episode.

Fig. 7. Rewards and savings of Case Study 2.

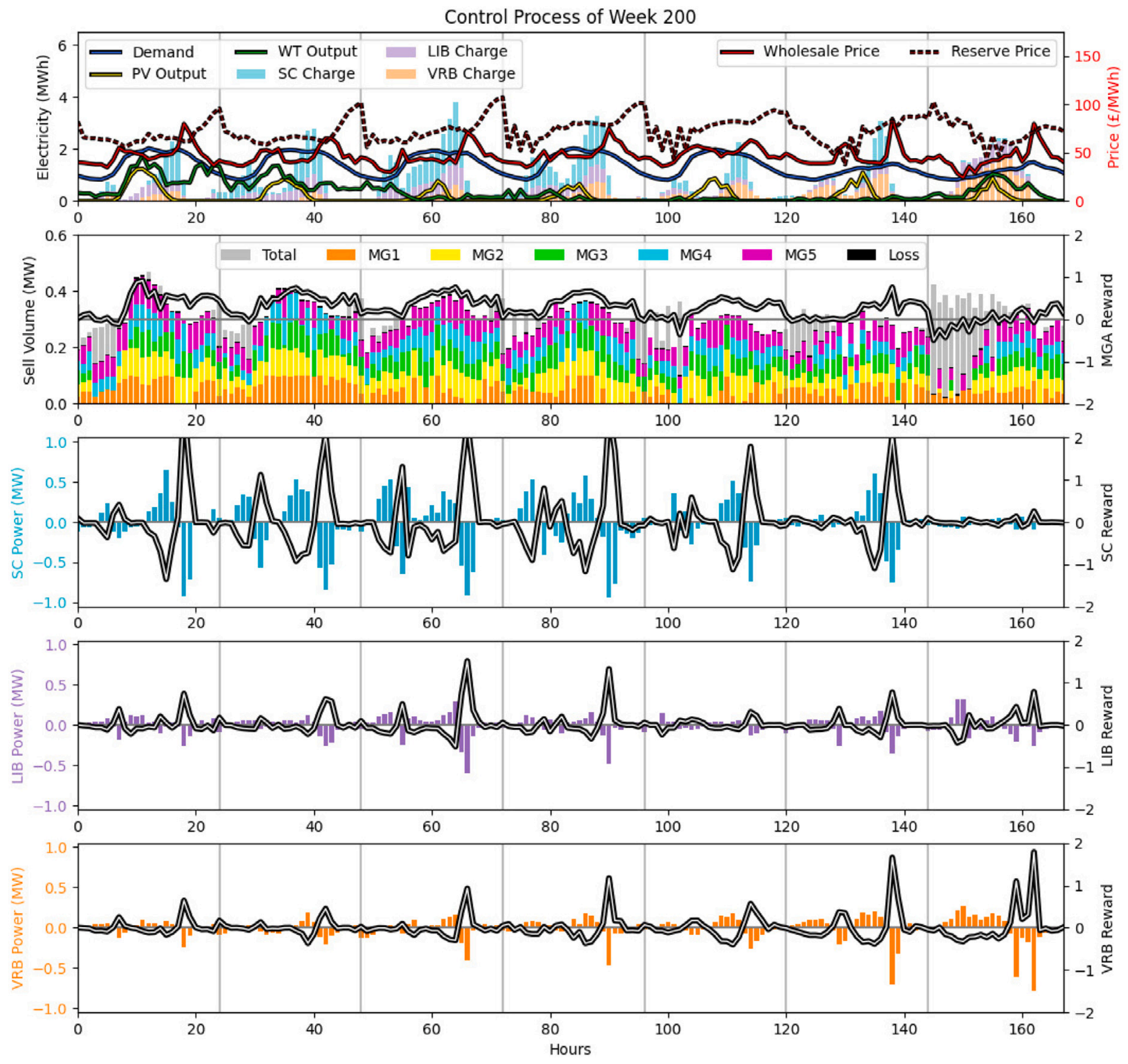


Fig. 8. Microgrid control in Episode 200 using MATD3.

own terms greatly increases savings over simply trying to improve its utilisation. The multi-agent methods once again perform better than the single controller, however the margin difference is significantly smaller. An example control process across one week is shown in Fig. 8.

5.2.1. Algorithm performance

The single-agent methods have a lower adjusted energy cost savings than the multi-agent methods, but interestingly D3PG and TD3 returned a lower ESS loss percentage and a higher MGA profit than any of the

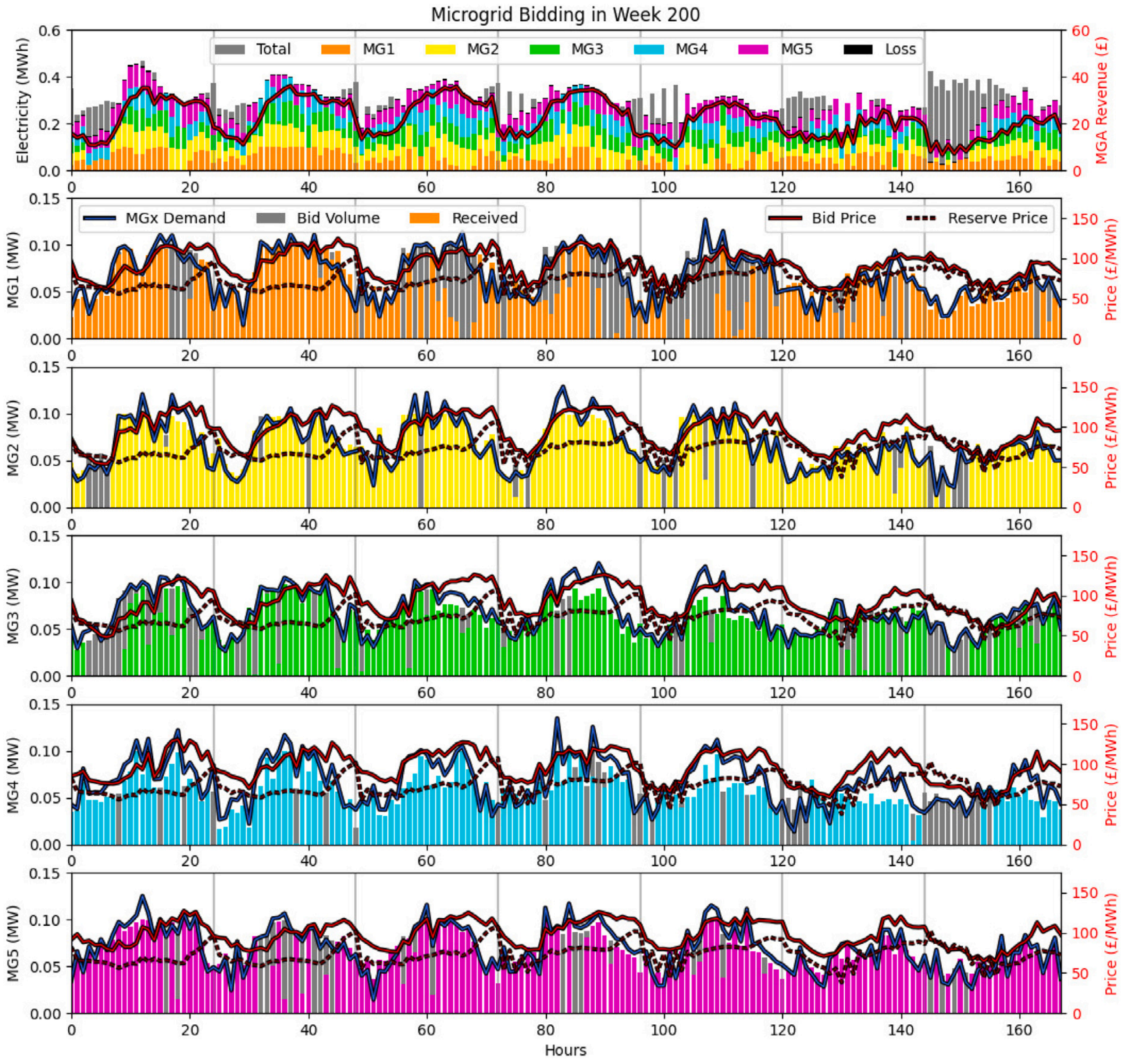


Fig. 9. MGA and xMG trading in Episode 200 using MATD3.

Table 5
Results of Case Study 1, with the best result of each column highlighted.

Algorithm	Case Study 2					
	Sav.(£1k)	Adj.(£1k)	vs DDPG	ESS Loss	MGA	MGA%
DDPG	151.38	146.46	–	3.25%	129.17	88.19%
D3PG	158.35	156.75	7.03%	**1.01%**	154.39	**98.49%**
TD3	174.04	170.63	16.50%	1.96%	**157.68**	92.41%
MADDPG	170.36	164.86	12.56%	3.23%	124.72	75.65%
MAD3PG	187.01	180.17	23.02%	3.66%	135.84	75.40%
MATD3	**190.54**	**186.00**	**27.00%**	2.38%	144.68	77.78%

Table 6
Adjusted savings of the different approaches.

Algorithm	Case Study 2		
	SGC	MAS-G	MAS-MC
DDPG	146.46	128.89	164.86
D3PG	156.75	154.56	180.17
TD3	170.63	182.21	**186.00**

multi-agent methods. A key difference between the approaches is that single global controllers try to exploit MGA trading significantly more than the multi-agent approach. D3PG again shows unique behaviour in that it almost entirely ignores the ESSs and instead makes 98.49% of its savings through MGA trading, significantly higher than the 77.78% of MATD3. Although interesting that D3PG and TD3 are able to make so much money through trading, this could be a cause for concern as the agents appear to have little interest in exploring how the HESS can be used in conjunction with trading for even higher savings.

This case study also shows the value of using the marginal contribution reward for the multi-agent methods, shown in Table 6. However, MATD3 using the MAS-G reward is still able to perform remarkably well with a noteworthy improvement over the single agent and only slightly worse than with the MAS-MC reward.

In fact, MATD3 performed the best out of all algorithms with a 27.00% improvement over DDPG. Interestingly, TD3 also performed very well relative to the other single agent methods and was also able to outperform MADDPG. This suggests that the value overestimation correction from TD3 is particularly useful in this case study. This is likely because, even though the agents are aware that they will receive a much higher reward through trading than through effective RES utilisation, the TD3 agents do not fixate entirely on trading and develop a suboptimal policy where they but instead recognise that effectively using the HESS to effectively utilise RES can enhance performance as well.

5.2.2. Agent behaviour

For the ESSs, SC is used more aggressively for short-term storage, LIB is used for medium-term storage, and the VRB has varied usage between used not at all to usage at the same level as the LIB. In general agents are less aggressive with RES utilisation as any excess RES can be sold rather than just stored, but also the multi-agent methods are more aggressive with the ESSs even when there is little RES generation. Some single-agent methods appear to even completely neglect the ESS operation for large parts of the simulation.

Initially, all agents take random actions for 1000 steps but begin learning after 500. Afterwards, the first observable behaviour comes from the xMGs which immediately learn that buying energy from the MGA is cheaper than buying from the utility grid so attempt to buy as much as possible. However, the MGA agent would learn one of two early behaviours:

1. Sell as much as possible at the minimum reserve price, such as MAD3PG and TD3. This leads to a high reward initially but the xMGs quickly learn to exploit this and lower their bid prices until the MGA ends up selling at lower than the wholesale price and losing money.
2. Sell nothing at the maximum reserve price, such as DDPG and MATD3. The MGA will eventually unlearn this behaviour, however the superior learning properties of TD3 over regular DDPG are clear as the MATD3 agent unlearns this significantly quicker than the DDPG and MADDPG agents.

After about 100 episodes, the agents reach an equilibrium and behave in a similar way but at slightly different levels of efficiency. Although the MGA could always sell an amount of energy that would meet the demands of all of the xMGs together, it will instead typically restrict the volume it sells to force the xMGs to compete with each other. An example of the trading is shown in Fig. 9.

As in the previous case study, the multi-agent approaches perform better than the single agent controllers. The separate reward functions for the different ESSs allows for each to learn their own policy suited to their characteristics while also allowing those agents to not become fixated on the high MGA rewards that the single agents fell short to.

This case study exhibits the benefit of being able to sell energy on the primary microgrid's terms, rather than only selling back to the utility grid. Trading is beneficial to all parties in that the primary microgrid is able to generate additional revenue, the xMGs are able to reduce their own energy bills, and inturn minimises the load on the utility grid.

5.3. Improvements and further work

One improvement would be expanding the functionality of the xMGs. This would include having their own ESSs and RES but also their own aggregator agents that would be able to trade with the other xMGs. This would take away much of the MGA's influence on the xMG trading phase and could allow the xMGs to develop their own unique behaviours.

A flaw in the environment is that wholesale energy prices are used from a day-ahead market, rather than a real-time market. To be more realistic, the primary microgrid should decide the volume of energy it wishes to import from the utility grid at least 24 h ahead. A more accurate feed-in tariff model based which varies based on the amount and type of energy sold could also be considered. However, the agents already learn to minimise the amount of energy sold back to the utility grid so would likely have a negligible impact on results or their behaviour.

Another improvement is forecasting further into the future for more efficient long-term storage. In most cases across both case studies, the agents tend to avoid using the VRB due to the poor cycling efficiency, but it would be interesting to see if the agents would change this behaviour if there were more forecasts provided so that the long-term storage properties of the VRB could be exploited to more effect.

Further work could look into other methods of communication for multi-agent RL. For example, differential inter-agent learning [56] allows the principle of centralised learning to be applied to DQN and Rainbow, rather than the current approach where the other agents are treated as part of the environment. There could also be other types of agents in the grid, such as a specific agent for purchasing energy from a day-ahead market or an agent controlling a distributed generator powered by diesel or biofuel.

6. Conclusion

In this paper, the use of multi-agent reinforcement learning was presented for the control of a HESS in a microgrid to increase renewable energy utilisation, reduce energy bills, and trade energy to xMGs. Specifically, the principle of centralised learning and decentralised execution with MADDPG is explored where agents can learn their own policies but evaluate their performance by considering the policies of every agent in the MAS, as well as more advanced variants of DDPG in D3PG and TD3.

The research found that the multi-agent approaches where each agent, with its their own reward function, controls an ESS performed better than a single global agent controlling the entire network. The separate reward functions for each component allowed the agents to more effectively evaluate their individual contribution to the shared goal, following the principle of marginal contribution from game theory. It was also found that selling energy to xMGs through the aggregator was significantly more profitable than simply maximising RES utilisation or selling back to the utility grid at a fixed feed-in tariff. Trading energy between the agents was beneficial to the primary microgrid through increased revenue, allowed the xMGs to reduce their own energy bills, and reduced the load on the main utility grid.

CRediT authorship contribution statement

Daniel J.B. Harrold: Conceptualization, Methodology, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Jun Cao:** Conceptualization, Validation, Formal analysis, Writing – review & editing, Supervision. **Zhong Fan:** Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Smart Energy Network Demonstrator project, United Kingdom (grant ref. 32R16P00706) funded by ERDF and BEIS, United Kingdom. This work is also supported by the EPSRC, United Kingdom EnergyREV project (EP/S031863/1) and the State Key Laboratory of Alternate Electrical Power System with Renewable Energy Sources, China (Grant No. LAPS20012).

References

[1] Ritchie H, Roser M. CO2 and greenhouse gas emissions. 2020, Our World in Data URL https://ourworldindata.org/emissions-by-sector.
[2] Bogdanov D, Ram M, Aghahosseini A, Gulagi A, Oyewo AS, Child M, et al. Low-cost renewable electricity as the key driver of the global energy transition towards sustainability. Energy 2021;227:120467. http://dx.doi.org/10.1016/j.energy.2021.120467.
[3] Ram M, Child M, Aghahosseini A, Bogdanov D, Lohrmann A, Breyer C. A comparative analysis of electricity generation costs from renewable, fossil fuel and nuclear sources in G20 countries for the period 2015–2030. J Cleaner Prod 2018;199:687–704. http://dx.doi.org/10.1016/j.jclepro.2018.07.159.
[4] Elia A, Kamidelivand M, Rogan F, Gallachóir BÓ. Impacts of innovation on renewable energy technology cost reductions. Renew Sustain Energy Rev 2021;138:110488. http://dx.doi.org/10.1016/j.rser.2020.110488.
[5] Jing W, Lai CH, Wong WS, Wong MD. A comprehensive study of battery-supercapacitor hybrid energy storage system for standalone PV power system in rural electrification. Appl Energy 2018;224:340–56. http://dx.doi.org/10.1016/j.apenergy.2018.04.106.
[6] Aneke M, Wang M. Energy storage technologies and real life applications – A state of the art review. Appl Energy 2016;179:350–77. http://dx.doi.org/10.1016/j.apenergy.2016.06.097.
[7] Vazquez S, Lukic SM, Galvan E, Franquelo LG, Carrasco JM. Energy storage systems for transport and grid applications. IEEE Trans Ind Electron 2010;3881–95.
[8] Sutton RS, Barto AG. Reinforcement learning: an introduction. Adaptive computation and machine learning series, 2nd ed.. Cambridge, MA: The MIT Press; 2018.
[9] Wooldridge M. An introduction to multiagent systems. John Wiley & Sons; 2009, Google-Books-ID. X3ZQ7yeDn2IC.
[10] Buşoniu L, Babuška R, De Schutter B. Multi-agent reinforcement learning: An overview. In: Kacprzyk J, Srinivasan D, Jain LC, editors. Innovations in multi-agent systems and applications. Vol. 310. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010, p. 183–221, Series Title: Studies in Computational Intelligence.
[11] Perera A, Kamalaruban P. Applications of reinforcement learning in energy systems. Renew Sustain Energy Rev 2021;137:110618. http://dx.doi.org/10.1016/j.rser.2020.110618.
[12] Vázquez-Canteli JR, Nagy Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89.
[13] Kuznetsova E, Li Y-F, Ruiz C, Zio E, Ault G, Bell K. Reinforcement learning for microgrid energy management. Energy 2013;59:133–46. http://dx.doi.org/10.1016/j.energy.2013.05.060.
[14] François-Lavet V, Taralla D, Ernst D, Fonteneau R. Deep reinforcement learning solutions for energy microgrids management. In: European workshop on reinforcement learning. 2016.
[15] Qiu X, Nguyen TA, Crow ML. Heterogeneous energy storage optimization for microgrids. IEEE Trans Smart Grid 2016;7(3):1453–61. http://dx.doi.org/10.1109/TSG.2015.2461134.
[16] Zhang G, Hu W, Cao D, Liu W, Huang R, Huang Q, et al. Data-driven optimal energy management for a wind-solar-diesel-battery-reverse osmosis hybrid energy system using a deep reinforcement learning approach. Energy Convers Manage 2021;227:113608.
[17] Prodan I, Zio E. A model predictive control framework for reliable microgrid energy management. Int J Electr Power Energy Syst 2014;61:399–409. http://dx.doi.org/10.1016/j.ijepes.2014.03.017.
[18] Li X, Dong C, Jiang W, Wu X. An improved coordination control for a novel hybrid AC/DC microgrid architecture with combined energy storage system. Appl Energy 2021;292. http://dx.doi.org/10.1016/j.apenergy.2021.116824.
[19] Foruzan E, Soh L, Asgarpoor S. Reinforcement learning approach for optimal distributed energy management in a microgrid. IEEE Trans Power Syst 2018;33(5):5749–58. http://dx.doi.org/10.1109/TPWRS.2018.2823641.
[20] Kofinas P, Dounis A, Vouros G. Fuzzy Q-learning for multi-agent decentralized energy management in microgrids. Appl Energy 2018;219:53–67. http://dx.doi.org/10.1016/j.apenergy.2018.03.017.
[21] Mbuwir BV, Geysen D, Spiessens F, Deconinck G. Reinforcement learning for control of flexibility providers in a residential microgrid. IET Smart Grid 2019;3(1):98–107. http://dx.doi.org/10.1049/iet-stg.2019.0196.
[22] Wang Y, Qiu D, Strbac G. Multi-agent deep reinforcement learning for resilience-driven routing and scheduling of mobile energy storage systems. Appl Energy 2022;310:118575. http://dx.doi.org/10.1016/j.apenergy.2022.118575.
[23] Li J, Yu T, Zhang X. Coordinated load frequency control of multi-area integrated energy system using multi-agent deep reinforcement learning. Appl Energy 2022;306:117900. http://dx.doi.org/10.1016/j.apenergy.2021.117900.
[24] Xu Y, Yu L, Bi G, Zhang M, Shen C. Deep reinforcement learning and blockchain for peer-to-peer energy trading among microgrids. In: 2020 International conferences on internet of things (ithings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData) and IEEE congress on cybermatics (Cybermatics). 2020, p. 360–5. http://dx.doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00071.
[25] Li W, Tang M, Zhang X, Gao D, Wang J. Operation of distributed battery considering demand response using deep reinforcement learning in grid edge control. Energies 2021;14(22):7749. http://dx.doi.org/10.3390/en14227749.
[26] Katiraei F, Iravani R, Hatziargyriou N, Dimeas A. Microgrids management. IEEE Power Energy Mag 2008;6(3):54–65. http://dx.doi.org/10.1109/MPE.2008.918702.
[27] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning. 2016, [Cs, Stat] arXiv:1509.02971.
[28] Watkins CJCHW. Learning from delayed rewards (Ph.D. thesis), UK: University of Cambridge; 1989.
[29] Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen RY, Chen X, et al. Parameter space noise for exploration. 2018, [Cs, Stat] arXiv:1706.01905.
[30] Fortunato M, Azar MG, Piot B, Menick J, Osband I, Graves A, et al. Noisy networks for exploration. 2018, [Cs, Stat] arXiv:1706.10295.
[31] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33.
[32] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, et al. Asynchronous methods for deep reinforcement learning. 2016, [Cs] arXiv:1602.01783.
[33] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, [Cs] arXiv:1707.06347.
[34] Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic policy gradient algorithms. In: International conference on machine learning. 2014, p. 387–95.
[35] Bellemare MG, Dabney W, Munos R. A distributional perspective on reinforcement learning. 2017, [Cs, Stat] arXiv:1707.06887.
[36] Barth-Maron G, Hoffman MW, Budden D, Dabney W, Horgan D, TB D, et al. Distributed distributional deterministic policy gradients. 2018, [Cs, Stat] arXiv:1804.08617.
[37] Thrun S, Schwartz A. Issues in using function approximation for reinforcement learning. In: Proceedings of the 1993 connectionist models summer school. 1993, p. 9.
[38] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: AAAI conference on artificial intelligence. 2016, p. 2094–100.
[39] van Hasselt H. Double Q-learning. In: Advances in neural information processing systems. Vol. 23. Curran Associates, Inc. 2010, p. 2613–21.
[40] Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods. 2018, [Cs, Stat] arXiv:1802.09477.
[41] Matignon L, Laurent GJ, Le Fort-Piat N. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. Knowl Eng Rev 2012;27(1):1–31. http://dx.doi.org/10.1017/S0269888912000057, Publisher: Cambridge University Press (CUP).
[42] Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. 2020, [Cs] arXiv:1706.02275.
[43] Rashid T, Samvelyan M, de Witt CS, Farquhar G, Foerster J, Whiteson S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. 2018, [Cs, Stat] arXiv:1803.11485. URL http://arxiv.org/abs/1803.11485.
[44] Foerster JN, Farquhar G, Afouras T, Nardelli N, Whiteson S. Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 32. 2018, p. 9.
[45] Harrold DJ, Cao J, Fan Z. Data-driven battery operation for energy arbitrage using rainbow deep reinforcement learning. Energy 2022;238:121958. http://dx.doi.org/10.1016/j.energy.2021.121958.
[46] Osborne MJ, Rubinstein A. A course in game theory. Cambridge, Mass: MIT Press; 1994.
[47] Carrillo C, Obando Montaño A, Cidrás J, Díaz-Dorado E. Review of power curve modelling for wind turbines. Renew Sustain Energy Rev 2013;21:572–81.
[48] Driesse A, Jain P, Harrison S. Beyond the curves: Modeling the electrical efficiency of photovoltaic inverters. In: 2008 33rd IEEE photovoltaic specialists conference. 2008, p. 1–6. http://dx.doi.org/10.1109/PVSC.2008.4922827.
[49] Nord Pool. Historical market data. 2020, URL https://www.nordpoolgroup.com/historical-market-data/.
[50] UK Power. Compare gas and electricity prices per kWh. 2021, URL.
[51] GOVUK. Feed-in tariffs: get money for generating your own electricity. 2021, GOV.UK URL.
[52] Ofgem. Feed-in tariff (FIT): Tariff table 1 April 2021. 2021, Ofgem URL https://www.ofgem.gov.uk/publications/.
[53] Boyd SP, Vandenberghe L. Convex optimization. Cambridge, UK, New York: Cambridge University Press; 2004.
[54] Tampuu A, Matiisen T, Kodelja D, Kuzovkin I, Korjus K, Aru J, et al. Multiagent cooperation and competition with deep reinforcement learning. In: Xia C-Y, editor. PLoS One 2017;12(4):e0172395. http://dx.doi.org/10.1371/journal.pone.0172395.
[55] Harrold DJB, Cao J, Fan Z. Battery control in a smart energy network using double dueling deep Q-networks. In: 2020 IEEE PES innovative smart grid technologies Europe. 2020, p. 106–10. http://dx.doi.org/10.1109/ISGT-Europe47291.2020.9248785.
[56] Foerster JN, Assael YM, de Freitas N, Whiteson S. Learning to communicate with deep multi-agent reinforcement learning. 2016, [Cs] arXiv:1605.06676.