Alexandria Engineering Journal (2022) 61, 9069–9078

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

Microgrid energy management using deep Q-network reinforcement learning

Mohammed H. Alabdullah [a,b], **Mohammad A. Abido** [b,c,d,*]

[a] *Saudi Aramco, Dhahran, Saudi Arabia*
[b] *Electrical Engineering Department, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia*
[c] *KACARE Energy Research & Innovation Center (ERIC), KFUPM, Saudi Arabia*
[d] *Interdisciplinary Research Center in Renewable Energy and Power Systems (IRC-REPS), KFUPM, Saudi Arabia*

Received 20 November 2021; revised 21 January 2022; accepted 13 February 2022
Available online 28 February 2022

KEYWORDS

Deep reinforcement learning;
Deep Q-networks;
Energy management;
Microgrid

Abstract This paper proposes a deep reinforcement learning-based approach to optimally manage the different energy resources within a microgrid. The proposed methodology considers the stochastic behavior of the main elements, which include load profile, generation profile, and pricing signals. The energy management problem is formulated as a finite horizon Markov Decision Process (MDP) by defining the state, action, reward, and objective functions, without prior knowledge of the transition probabilities. Such formulation does not require explicit model of the microgrid, making use of the accumulated data and interaction with the microgrid to derive the optimal policy. An efficient reinforcement learning algorithm based on deep Q-networks is implemented to solve the developed formulation. To confirm the effectiveness of such methodology, a case study based on a real microgrid is implemented. The results of the proposed methodology demonstrate its capability to obtain online scheduling of various energy resources within a microgrid with optimal cost-effective actions under stochastic conditions. The achieved costs of operation are within 2% of those obtained in the optimal schedule.

1. Introduction

A microgrid is defined as a group of loads and micro-sources operating under the control of one system [1]. The microgrid could operate in parallel with the utility grid to optimally consume local power generation sources, or in islanded mode in case of failure in the main grid, thus enhancing overall reliability of local service. Many benefits are realized by adopting microgrids, including the reduction of greenhouse gases emission, improving voltage profiles, decentralization of power supply and reducing line losses [2]. It also allows customers to actively participate in the microgrid operation [3].

The decrease of renewable generation costs has driven the adoption of microgrid schemes. For example, the cost of manufacturing solar PV has seen noticeable reduction over the past years, which was accompanied by a huge increase in installa-

* Corresponding author.
E-mail addresses: mohammed.alabdullah@aramco.com (M.H. Alabdullah), mabido@kfupm.edu.sa (M.A. Abido).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2022.02.042

Nomenclature

$P_{min}^{DG_d}$	The minimum active power output of generator d
$P_{max}^{DG_d}$	The maximum active power output of generator d
$S_{max}^{DG_d}$	The maximum apparent power output of generator d
$Q_t^{DG_d}$	The reactive power output of generator d, at time t
$C_t^{DG_d}$	The cost of operating conventional generator at time t
Δt	Time period
P_t^E	Charging or discharging power of energy storage device at time t
P_{max}^E	The maximum charging/discharging power of energy storage device
E_t	Energy level of energy storage device or state of charge
E_{min}	Minimum energy level of energy storage device
E_{max}	Maximum energy level of energy storage device
η_{ch}	Charging efficiency of energy storage device
η_{dis}	Discharging efficiency of energy storage device
P_t^U	Active Power exchange of microgrid with grid at time t
P_{max}^U	Maximum active power exchange of microgrid with grid at time t
Q_t^U	Reactive power exchange of microgrid with grid at time t
S_{max}^U	Maximum complex power exchange of microgrid with grid at time t
C_t^U	Cost of purchase of electricity from grid at time t
R_t	Price of electricity from grid at time t
P_t^{PV}	Power output from solar PV at time t
P_t^W	Power output from wind at time t
Θ	Neural Network parameters
α	Learning rate
∇	Gradient
$\mathscr{S}$	State Space
R	Reward function
Y	Discount factor
π	RL policy
$Q^\pi(s,a)$	State-action value function that follows policy π

tions. As a matter of fact, the global solar PV energy capacity grew nearly ten times within one decade, from 72 GW in 2011 to more than 707 GW in 2020 [4]. Nonetheless, there are several challenges to integrate renewable distributed generation resources, which mainly stem from their intermittent nature, making it difficult to optimally schedule generation as practiced in a conventional grid. Unforeseen power variations would necessitate the commitment of expensive reserve or ancillary services, making the microgrid operation uneconomical.

Many studies have been conducted to overcome such challenge. For instance, [5] and [6] present a two-stage stochastic programming methodology to optimize the operation costs. Also, [7] considers different cost variations and adopts a risk-averse stochastic programming method. However, the aforementioned papers require information of the uncertainty statistical distribution beforehand. To work around the unavailability of statistical distributions, [8,9] use robust optimization to derive an optimal solution from possible worst-case conditions. However, these studies derive fixed schedules that do not respond to unplanned variations in real time.

The use of energy storage systems (ESS) can mitigate the issues of matching generation and demand variations. ESS allow the system operator to have more flexibility over the microgrid resources, and to shift the intermittent renewable generation to peak hours, thus earning from energy arbitrage [10]. Many other benefits can be realized by having ESS, which include providing ancillary services, such as load following, voltage support and frequency regulation [11]. Additionally, the use of energy storage systems helps increasing the reliability of power delivered to customers during disturbances from supply side [12]. However, ESS management is complex as it is not operated like conventional generation by making economic dispatch or unit commitment. Efficient operation of ESS requires an energy management system (EMS) that maximizes the operation benefits of distributed generation with energy storage and considers long-term time varying prices, demands, and renewable generations [13].

In conventional methods, the microgrid energy management problem is generally solved using a model-based framework to formulate the dynamics of the microgrid. Then, the uncertainties are estimated using a predictor, and the optimal schedule is obtained using an optimization problem solver [14,15]. Examples include the use of rolling horizon method [16], in which a mixed integer optimization formulation is solved for each decision step. Additionally, [17] develops a convex Model Predictive Control (MPC) methodology for dynamic optimal power flow in multiple battery storage systems in a microgrid. Others [18,19] designed a hierarchical control structure to integrate the operation management of multiple interconnected microgrids, with a central controller overseeing all microgrids, and secondary controllers which implement different model predictive strategies to manage the local operation.

Although such methods proved their effectiveness in the energy management domain, they highly depend on an expert to accurately model the dynamics of the microgrid. Since the scale, capacity, and dynamics of the microgrid could change with time, the uncertainty profiles of generation and demand will change significantly [20]. This may limit the applicability of these methods with large-scale microgrid energy problem in real-time. Moreover, building a generalized framework using these methods for different environments is a challenging task [21]. This leads to increasing the complexity and cost associated with implementing EMS for different microgrids, which could hinder the adoption of such grid schemes.

To work around this problem, learning based methodologies have been proposed in the literature to resolve microgrid management difficulties. Such methods do not require explicit model of the microgrid, making use of the accumulated data and interaction with the system to derive the optimal control policy [22]. For example, [23] presents an energy management

system using batch reinforcement learning. Additionally, [24] develops a solution to microgrid energy management using evolutionary adaptive dynamic programming and reinforcement learning. Also, [25] uses reinforcement learning framework to design a multiagent system that aims to optimize the microgrid energy management. Although, aforementioned works present novel ways to handle the energy management problem, they struggle with high dimensional state variables due the curse of dimensionality.

Alternatively, deep reinforcement learning (DRL) has been introduced in the literature in recent years [26], which tackle the problem of high dimensional state spaces. It makes use of deep neural network to extract important features of the high dimensional state as the agent is exposed to experiences from the interaction with the environment, with no prior knowledge of the system. The agent does not need a separate model to obtain a forecast or a probability distribution to the changing variables. Rather, the agent makes full use of the available set of collected data to learn the optimal policy that achieves optimal results. The deep reinforcement learning methodologies can be implemented to achieve autonomous real time control and decision making in different power system applications. For example, [27] applied DRL in strategic bidding and equilibrium analysis in electricity markets. Also, [28] adopted DRL to power management in distribution systems with bi-level power management of cooperatives that has several networked microgrids. Reference [29] applied DRL to optimize the demand side peak to average ratio in multi microgrids. Moreover, reference [30] developed an adaptive emergency control scheme based on DRL. Other applications include voltage control [31], control for multi energy systems [20] and others.

In terms of energy management applications, several works have been reported in the literature that make use of DRL. For example, [32] formulates the microgrid management as a sequential decision-making problem under uncertainty. A convolutional NN is used to extract important features and output Q-values that are used by the reinforcement policy. The study proved effective in a case study, though, it only considered a simple network with no constraints such as power flows, or real time electricity prices. Additionally, [13] devises a method for real time economical operation of a microgrid using approximate dynamic programming. It considers system constraints and uncertainties in the load and generation as well as a variable pricing of electricity. A deep recurrent neural network is used to predict the changing variables, which in turn is used to obtain the value function approximation. However, the work is formulated to have a predictor to solve the formulation, which opposes the model-free paradigm. Reference [22] formulates deep Q-networks (DQN) to manage energy resources in a low voltage network. Though, the work considers a simple objective of minimizing costs due grid exchanges and conventional generation. It also ignores power flow constraints in the network, which will likely result in violating network constraints.

This paper considers a learning based methodology based on deep Q-networks to optimally manage the different energy resources in a realistic model of microgrids. The methodology considers the stochastic behavior of different elements of a microgrid, including loads, generations, and electric prices. It also models different grid elements using equivalent models and considers the various power flow constraints in a realistic setting.

The paper is organized as follows: section II briefly introduces the basis and formulation of deep reinforcement learning framework. Section III provides the problem formulation and implementation of the microgrid energy management problem in the context of deep reinforcement learning. A case study is presented in Section IV. Section V discusses the results obtained for the case study. Finally, section VI concludes the work and provides recommendations for future work.

2. Reinforcement learning

2.1. Overview

Reinforcement learning (RL) relates to sequential decision making in which an agent interacts with an environment so that cumulative reward signals are optimized. The interaction occurs in a trial–error basis, and the agent learns a good behavior from previous experiences [33]. Formally, the reinforcement learning can by described by a discrete time stochastic control process. An agent starts in a state $s_0 \in \mathscr{S}$, and makes an initial observation $\omega_0 \epsilon \Omega$. The agent makes an action at time step t, $a_t \in \mathscr{A}$. Then, the agent obtain a reward $r_t \in \mathscr{R}$, the state transitions to $s_{t+1} \in \mathscr{S}$, and a new observation is obtained $\omega_{t+1} \epsilon \Omega$, as shown in Fig. 1.

This formulation can be viewed as a Markov Decision Process (MDP) if the Markov property is satisfied. This entails that future states depend only on current observation, and no full history is required. A fully observable MDP means that the observation matches the state, $\omega_t = s_t$. An MDP is described by the following:

$\mathscr{S}$: state space

$\mathscr{A}$: action space

T: transition function $(T : \mathscr{S} \times \mathscr{A} \times \mathscr{S}) \rightarrow [0, 1]$

R: reward function $(R : \mathscr{S} \times \mathscr{A} \times \mathscr{S}) \rightarrow \mathscr{R}$, where $\mathscr{R}$ is a continuous set of possible rewards

Y: discount factor, $\mathrm{Y} \in [0, 1)$

Under this formulation, an agent selects an action dictated by policy π. Such policy can be either stationary (time dependent), or non-stationary (time independent). It can also be either deterministic or stochastic. An RL agent aims to find a policy $\pi(s, a)$, so that the expected return is optimized. The return can be expressed as a Q-value function as follows:

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi\right] \tag{1}$$

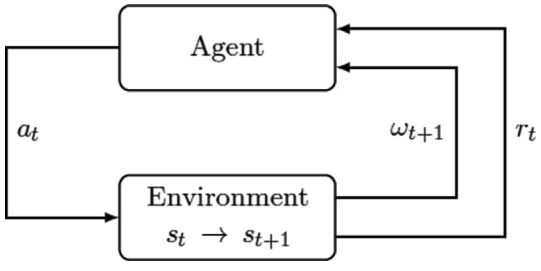

Fig. 1 Reinforcement learning process.

$r_t = \mathbb{E}[R(s_t, a, s_{t+1})]$

This equation can be expressed recursively (using Bellman's equation) as:

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma Q^{\pi}(s', a = \pi(s'))) \tag{2}$$

The optimal Q-value function $Q^*(s,a)$, can be expressed as:

$$Q^*(s,a) = \underbrace{\max}_{\pi \in \Pi} Q^{\pi}(s,a) \tag{3}$$

Additionally, the optimal policy can be defined as:

$$\pi^*(s) = \underbrace{\text{argmax}}_{a \in A} Q^*(s,a) \tag{4}$$

The optimal Q-value function $Q^*(s,a)$ represents the expected discounted return in state s, and following action a that is dictated by policy π^*. One straight way to obtain $Q^{\pi}(s,a)$ is to use Monte Carlo to estimate the values from many simulations following a given policy. Practically, this is not an efficient method to learn the value functions [34].

A simple Q-learning algorithm keeps a lookup table for each corresponding state and action. The optimal Q value is learned through the Bellman equation, but such method is inapplicable with a high-dimensional state-action space. Alternatively, the value function can be parametrized $Q(s,a;\theta)$, in which θ represents the parameters defining the Q values. This gives rise to variation of Q-learning algorithms such as DQN, which has proved its effectiveness and superior performance [26].

2.2. Deep reinforcement learning

Deep learning provides several advantages in the context of reinforcement learning. Many real practical problems are complex with a high dimensional continues state space, and so neural networks are well suited for high-dimensional inputs. They can also be trained incrementally and learn progressively as they are provided with more data. We discuss an off-policy DRL methodology based on Q-learning.

In fitted Q-learning [35], a dataset D is constructed using past experiences in the form of (s,a,r,s'), in which reward r, and next state s' follow state-action (s,a). The Q-value $Q(s,a;\theta_k)$ is updated at each iteration k to a target value Y_k, where:

$$Y_k = r + \gamma max_{a'} Q(s', a'; \theta_k) \tag{5}$$

θ_k represents the parameters that define the Q-values at iteration k.

Such formulation can be applied in a neural network, where states can be provided as inputs to the neural network, and the outputs give the Q-values for each state-action. The network parameters θ_k are updated using methods such as stochastic gradient descent which minimizes the square error:

$$\mathrm{L} = (Q(s,a;\theta_k) - Y_k)^2 \tag{6}$$

Hence, to update parameters at the next iteration, we apply the following:

$$\theta_{k+1} = \theta_k + \alpha(Y_k - Q(s,a;\theta_k))\nabla_{\theta_k} Q(s,a;\theta_k) \tag{7}$$

where α is a scaler value defining the learning rate. Iterating through this update process should in hypothesis converge to an optimal solution. However, such formulation could suffer from slow convergence and possible instability, mainly due to propagated errors from the neural network generalization property, and the simultaneous update of the target network parameters [34].

To bridge the gaps found in aforementioned Q-learning algorithms, deep Q-learning network (DQN) presented by [36] makes use of a separate target network with parameters θ_k^-, that are updated every C iterations with θ_k parameters. This method results in preventing divergence and reducing instabilities of learnt Q values. This is represented in the following equation:

$$Y_k = r + \gamma max_{a'} Q(s', a'; \theta_k^-) \tag{8}$$

Additionally, a reply memory is used to keep the last N_{replay} steps experiences in the form (s,a,r,s') . Such that it spans across the state action space, and have less variance as opposed to making a single step update. The algorithm is illustrated in Fig. 2. To further illustrate the steps needed to implement DQN, we provide algorithm 1, which provides a pseudo code for DQN as presented in [36].

Input: Microgrid observed states (e.g. SOC, Load MW, Generation MW, etc.)

Output: Q action values that determine policy to selection action (e.g. controlling energy resources)

Initializing replay memory D

Initialize Q-network with random weights θ

Initialize target Q-network with weights $\theta^- = \theta$

For $episode = 1 to M$ **do**:

Initialize sequence $s_1 = \{x_1\}$

For $t = 1 to T$ **do**:

While following ϵ-greedy policy, choose action

$$a_t = \begin{cases} random action with probablity \epsilon \\ argmax_a Q(s_t, a; \theta) otherwise \end{cases}$$

Execute action a_t in environment and observe reward r_t and next state s_{t+1}

Store transition (s_t, a_t, r_t, s_{t+1}) in replay memory D

Sample random batches of transitions s_k, a_k, r_k, s_{k+1} from D

Set $y_k = \begin{cases} r_k if terminal state is reached \\ r_k + \gamma max_{a'} Q(s_{k+1}, a'; \theta^-) otherwise \end{cases}$

Perform gradient descent step on $(y_k - Q(s_k, a_k; \theta))^2$ with respect to q-network parameters θ

Set $\theta^- = \theta$ at every C steps

End

End

Algorithm 1: DQN algorithm implementation as per [36]

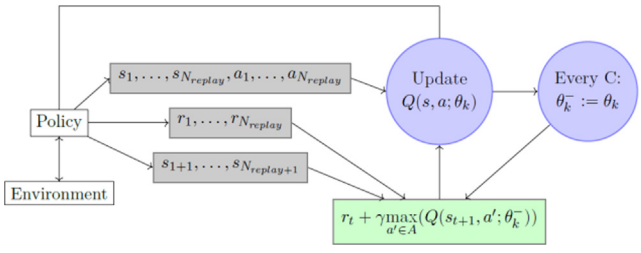

Fig. 2 DQN algorithm flowchart [34].

To further enhance the DQN algorithm convergence and stability, the following heuristics are considered.

2.2.1. E. Priority replay memory

Experience replay allows an RL agent to make use of past experiences. In the previous DQN algorithm, sampling occur uniformly from the replay memory, while ignoring how important the sampled transitions. Reference [37] developed a method to prioritize experience replays, so that significant transitions are more frequently replayed to learn more effectively. The method uses importance sampling based on temporal difference error to select significant transitions and thus enhance overall performance.

2.2.2. Double DQN

In the discussed DQN algorithm, the max operator is used to select and evaluate an action as detailed in algorithm 1. Consequently, overestimated values are likely to be selected. Reference [38] proposed double DQN (DDQN) to solve this overestimate issue. To do so, the paper proposed the evaluation of greedy policy according to the online network (with parameters θ_k) , while using the target network (θ_k^-) to estimate its value. The target value is then expressed as follows:

$$Y_k = r + \gamma Q\left(s', \arg\max_a Q(s', a; \theta_k); \theta_k^-\right) \tag{9}$$

2.2.3. Linear annealed exploration

As opposed to using a fixed low ϵ value while following an ϵ-greedy policy, we can start training using a high ϵ value, and then decrease it linearly as we continue training, arriving to a low ϵ value. Such way ensures the agent makes a lot of exploration earlier, while exploiting the accumulated information at the end [33].

3. Problem formulation

When it comes to developing the framework of the microgrid environment, there are plenty of options to consider. Several papers adopted different elements of the microgrid such as limitation on the demand side as well as constraints on the generation side [39]. Reference [40] considers operation constraints in the microgrid as well as limits in renewable generation. Other papers looked into energy storage constraints such as charging and discharging rates, limits on electricity prices, or carbon emissions [41]. This paper considers the most critical constraints of microgrid, namely constraints on the power exchange with external grid, power output constraints from different distributed generation devices, and power flow constraints. The adopted model of microgrid is interconnected with the utility at PCC, and has some conventional distributed generators, energy storage systems, solar PV, wind turbines, and variable loads.

The proposed formulation considers conventional generators that are constrained by the following equations:

$$P_{min}^{DG_d} \le P_t^{DG_d} \le P_{max}^{DG_d} \tag{10}$$

$$P_t^{DG_d 2} + Q_t^{DG_d 2} \le S_{max}^{DG_d 2} \tag{11}$$

In which $P_t^{DG_d}, Q_t^{DG_d}$ represent the active and reactive power output of generator d, at time t. Rating of the generator is given by $S_{max}^{DG_d}$. The cost function of operating conventional generators is given by a quadratic function as shown in the following equation:

$$C_t^{DG_d} = \left[a_d\left(P_t^{DG_d}\right)^2 + b_d P_t^{DG_d} + c_d\right]\Delta t \tag{12}$$

$a_d, b_d, and c_d$ are constants.

Additionally, energy storage devices are constrained by the following equations:

$$0 \le P_t^E \le P_{max}^E \tag{13}$$

$$E_{min} \le E_t \le E_{max} \tag{14}$$

$$E_t = E_{t-1} + \eta_{ch} u_t P_t^E \Delta_t - (1 - u_t) P_t^E \Delta_t / \eta_{dis} \tag{15}$$

In which the charging or discharging power is represented by P_t^E, and the energy level (SOC) is given by E_t. A binary variable u_t is used to indicate if the ESS is charging, $u_t = 1$, or discharging, $u_t = 0$. We use η_{ch} and η_{dis} to represent the charging/discharging efficiencies accordingly. The time period of charging/discharging is represented by Δt.

The power exchange with the utility is considered, governed by the following equations:

$$-P_{max}^U \le P_t^U \le P_{max}^U, \forall t \tag{16}$$

$$P_t^{U2} + Q_t^{U2} \le \left(S_{max}^U\right)^2 \tag{17}$$

In which P_t^U and Q_t^U represent the active/reactive power exchanges with the utility. The maximum exchange of complex power is given by S_{max}^U. The cost of purchasing power from the utility is given by the following equation, in which R_t is the real time price.

$$C_t^U = P_t^U \cdot R_t \cdot \Delta t \tag{18}$$

Additionally, power flow constraints are considered while controlling the energy resources. This is represented by power flow limits at each branch ij, as shown below:

$$P_t^{ij2} + Q_t^{ij2} \le {S_{max}^{ij}}^2 \tag{19}$$

To ensure voltage limits are within standard values, the voltage is constrained as follows:

$$\left|V^i\right|_{min} \le \left|V_t^i\right| \le \left|V^i\right|_{max} \tag{20}$$

where V_t^i is the voltage at bus i at time t is limited by given minimum and maximum values.

Using the aforementioned model of the microgrid, an MDP is formulated as follows. The state variables at each time step t is given by $\left(\boldsymbol{P_t^{PV}}, \boldsymbol{P_t^W}, \boldsymbol{P_t^D}, \boldsymbol{Q_t^D}, \boldsymbol{R_t}, \boldsymbol{E_t}\right)$, in which $\boldsymbol{P_t^{PV}}$ is the power output from the solar PV plant, $\boldsymbol{P_t^W}$ is the power output of the wind turbine, $\boldsymbol{P_t^D}$ *and* $\boldsymbol{Q_t^D}$ *are* the loads active and reactive powers, $\boldsymbol{R_t}$ is the real time price of electricity, and E_t is the energy level of the energy storage system. The model can be formulated to take only current state variables, or past historical variables.

The action space is given by $\left(P_t^{DG_d}, Q_t^{DG_d}, P_t^E\right)$, in which $P_t^{DG_d}, Q_t^{DG_d}$ are the active/reactive power from conventional generators, and P_t^E is the charging/discharging power of the

energy storage system. The action space is developed to be discrete in order to be compatible with the DQN algorithm.

In terms of reward function, the main objective is to optimize the operational costs while adhering to system constraints. Hence, the reward function is formulated to correlate with the microgrid operational costs. These include the costs of operating the conventional generators, and power purchases from utility. Equation (21) provides a simple formulation of the reward function at each time step.

$$r_t = -\left(\sum C_t^{DG_d} + C_t^U\right) \tag{21}$$

In addition, the reward function considers any violation to the power flow constraints, by providing a negative reward, and terminating the training episode. It also has a negative reward whenever an action violates energy storage constraints, so that violating actions are discouraged.

These formulation steps are summarized in Fig. 3, which shows a high-level flowchart of the different steps taken to solve the problem.

The interaction of agent with the environment, and the overall flow of training is illustrated in Fig. 4. The state variables of the microgrid are provided to the q-network, which in turn makes a decision based on ϵ-greedy policy that maximizes the q-value. The interactions between the environment and the agents are collected and saved in a replay memory, to be used for training the online network, and the target network. This process continues until episodes end, or termination conditions are met.

4. Case study

To study the proposed methodology, we implement a MV CIGRE microgrid as illustrated in Fig. 5. The different parameters used to implement this network can be found in [42]. The microgrid has different conventional and non-conventional

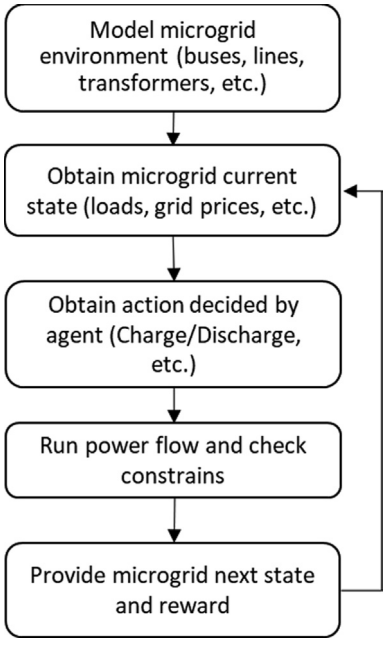

Fig. 3 Flowchart of proposed methodolgy.

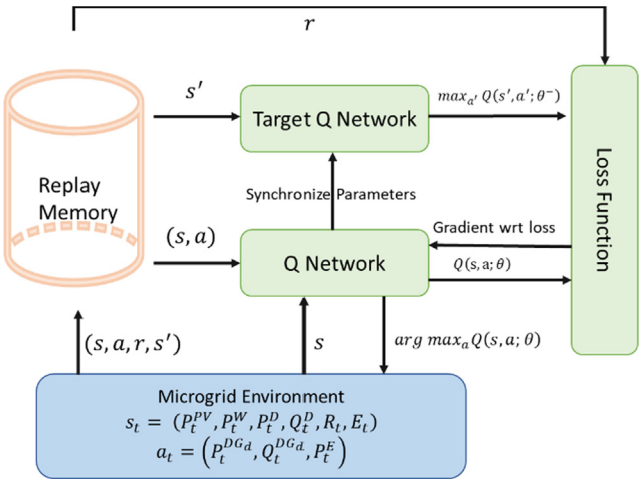

Fig. 4 Architecture of DQN for microgrid energy management.

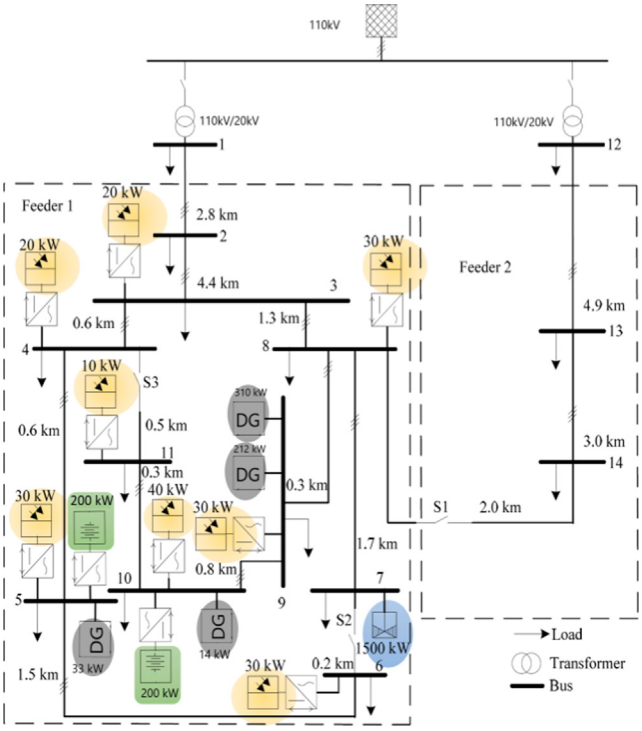

Fig. 5 CIGRE MV Microgrid.

distributed power generation units including PV, wind, diesel generators, and energy storage units. It is also connected to an external grid through a PCC.

The microgrid environment is set up using OpenAI Gym library [43] to provide RL interface framework. We use Pandapower toolbox [44] to run power flow calculations and ensure that constraints are not violated. Stable baselines [45] is used to implement the DQN agent to solve the set up environment.

A feedforward NN is used to map the different states at any given time to their corresponding state action value function for each action a_t^k. The network takes as an input the past 24 h values of the different variables (e.g. PV, wind, grid prices,

etc), and outputs the Q-Value for each possible action (e.g. charging/discharging, etc) at current state. Rectified linear units (ReLU) are used as the activation function in the hidden layers.

The target network is constructed with a similar structure to the training Q-network and is updated frequently. A replay memory of size 50,000 is constructed to store (s, a, r, s') transitions at each step. Initially, the agent is allowed to take random actions for the first 1000 steps before the learning starts. The performance of the training network is logged, with parameters being saved periodically. The parameters of the best performing network are saved to be used for testing.

5. Results & discussions

Several experiments are conducted to fine tune the NN different hyperparameters. Since training computation can take long hours (it takes around 24 h to run the training of 1 million steps with I7-8700 K CPU @ 3.7 GHz with 16 GB RAM), the various parameters are tuned manually, guided by intellectual intuition and literature findings. This shall not be an obstacle in real deployments, since advances in GPU and cloud computing can enhance this training time significantly.

Additionally, the date is divided in two sets: training (75%) and testing (25%). The agent is allowed to train in the developed framework for 500 K time steps in each network configuration. The results are then benchmarked to those obtained in the optimal case. While running the simulation on the test data, each time step takes around 0.01 s. Overall, it takes around 22 s for the 2160 h simulation steps. This demonstrates the ability of such method to conduct real time scheduling of energy resources.

5.1. Hyperparameter tuning

To find a suitable NN structure, simulations are conducted for different layers and number of nodes per hidden layers. It is observed that having two layers with at least 256 is efficient to reach acceptable reward levels. Figs. 6 & 7 show the learning curve for various number of NNs with two and three hidden layers. Using 3 layers is observed to have no advantage over 2 layers, and it might occasionally perform worse than two hidden layers. A larger network will result in less weight sharing, giving more learning stability. However, a larger network could result in more weights that needs to be changed, leading to deteriorating the policy performance.

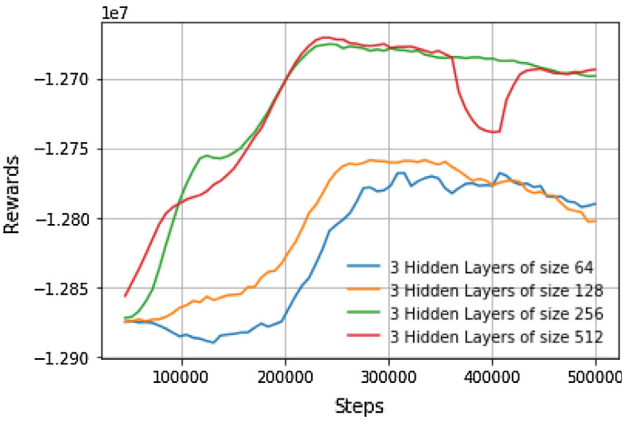

Fig. 7 Learning curve of NN with 3 hidden layers.

In terms of the tuning the learning rate, it was found that a learning rate between 0.001 and 0.0001 achieved best results. Moreover, we study the effect of the discount factor (γ) used in updating the target network value. The discount factor has a direct effect on the objective of the agent performance. Since a main part of the reward function depends on the stochastic electricity prices, which makes predicting the future harder. Hence, lowering discount factor is more beneficial in optimizing the reward accumulation. Fig. 8 shows the learning curves for different agents with different discount factors. Having gamma values below 0.5 performed much better than higher values. This means that current actions only affect shorter time periods in the future.

In terms of the effect of the size of batches used to apply gradient descent to update the training network weights, it is observed that the larger the batch size the better learning performance, though batch sizes larger than 256 had no significant improvements.

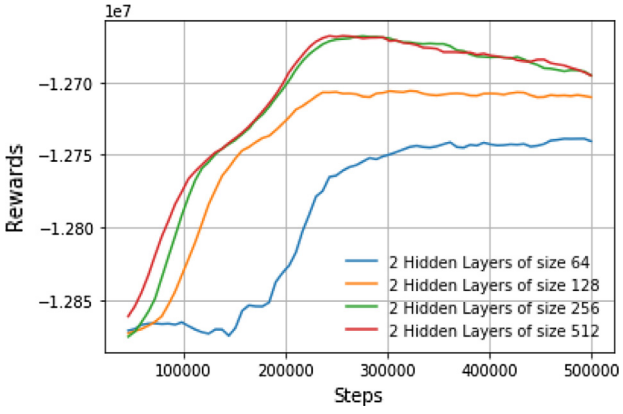

Fig. 6 Learning curve of NN with 2 hidden layers.

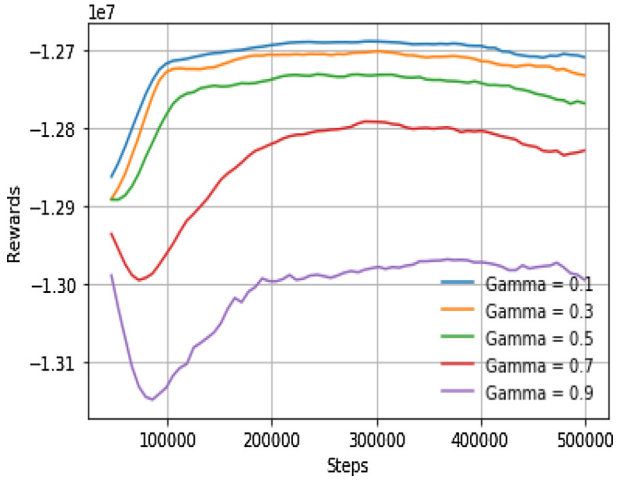

Fig. 8 Discount factor effect on training performance.

5.2. Energy schedules

In order to examine the performance of the developed agent, scheduling is simulated for the energy resources in the microgrid. The simulation is done in a test environment that uses historical data for 3 months. Fig. 9 shows the total charging/discharging schedules and SOC for the energy storage devices. The agent is allowed to charge/discharge the energy storage devices, as long as it does not violate specified dynamic constraints of the energy storage. The agent tries to maximize the benefits of controlling the energy storage by accounting for different variables including external grid electric prices, and available resources withing the microgrid. Fig. 10 shows the generation schedules for the diesel generators and energy storage devices, and external grids. Similar to energy storages, the agent tries to optimize the operation of the DG units taking into account the costs of operating the units, as well as other state variables.

To benchmark the performance of the developed agent, we compare the daily costs incurred by following the agent actions, and the optimal solution obtained by a mix integer linear programming methodology (MILP). As opposed to proposed methodology, the MILP solver has access to future variable values across the time horizon, and hence does not account for uncertainty. Fig. 11 shows the daily incurred costs for both solutions during the test period. To further show the differences, Fig. 12 shows the difference in daily costs between the optimal and DQN solution.

To illustrate the performance of the developed DQN method, we plot the accumulated costs along the test period in Fig. 13. We provide two DQN agents next to the optimal solution, one with untuned hyperparameters, and another with tuned hyperparameters. Overall, DQN showed its capability to conduct energy management, with results that are comparable to those obtained in the optimal solution. Additionally, when computing the actions in the testing set, it only took 0.01 s per step for the DQN agent, compared to 2.4 s with the MILP solver.

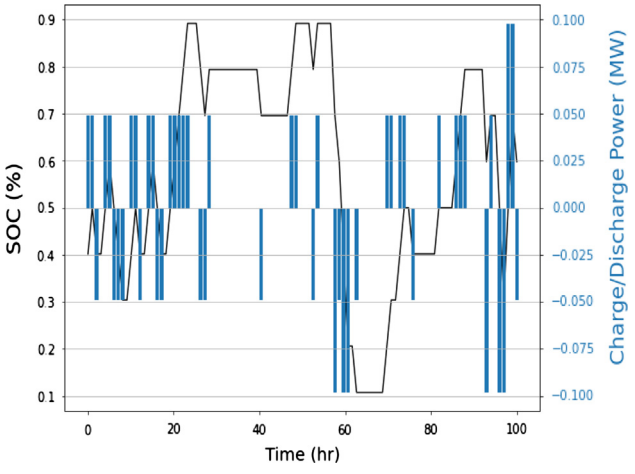

Fig. 9 Energy storage hourly schedule with SOC.

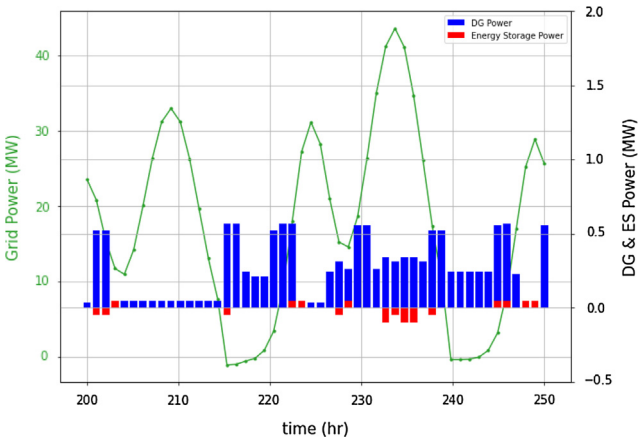

Fig. 10 DG and energy storage hourly schedules, and external grid power exchange.

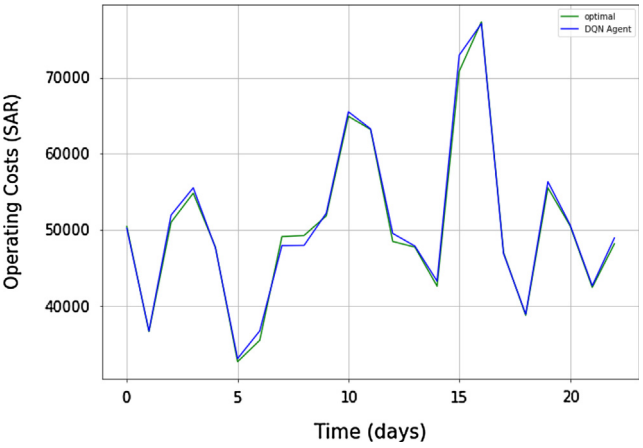

Fig. 11 Benchmarking DQN agent daily operating costs with optimal solution time horizon.

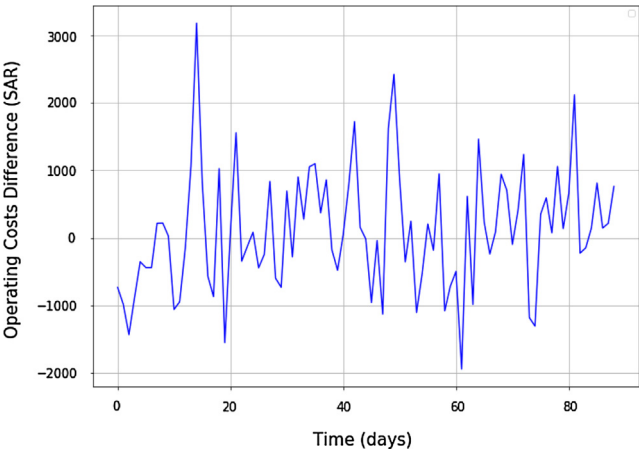

Fig. 12 Daily costs difference between optimal solution and DQN solution.

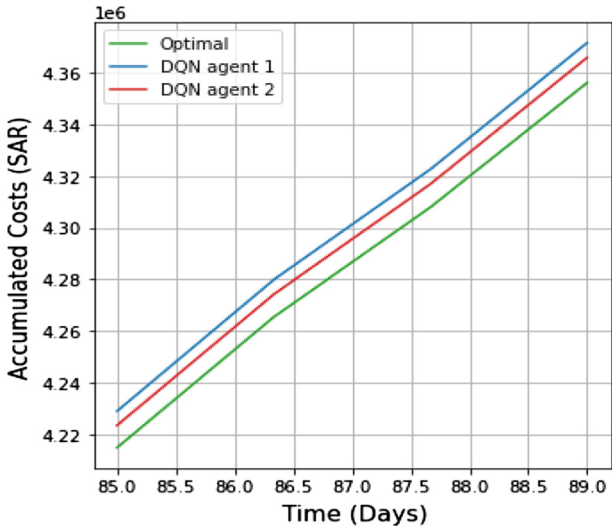

Fig. 13 Accumulated operating costs for optimal solution, untuned DQN agent, and tuned DQN agent.

6. Conclusion

This paper proposed the use of deep reinforcement learning methodology based on deep Q-network algorithm to solve the energy management problem formulation of a given microgrid. The methodology considered the stochastic behavior of different elements of a microgrid, and modeled different grid elements while adhering to the various power flow constraints in a realistic setting. Such formulation tackles some gaps that exist in conventional methods, such as dependability on experts to model the dynamics of the microgrid, achieving real-time scheduling, and providing a generalized framework for different environments. The developed framework can be adjusted for different microgrid architectures providing flexibility to test the set-up for different types of electrical grids. In this study, it was demonstrated that the deep reinforcement learning methodologies as implemented can obtain near optimal results. The methodology can conduct online scheduling of the various energy resources within a grid and make cost effective actions under stochastic conditions. The results were benchmarked with the optimal results obtained by MILP solver that had full knowledge of the different stochastic variables. The costs of operation achieved are within 2% of the optimal case scenario. Additionally, the computation time of the developed method is on average 0.01 s per step compared to 2.4 s with the MILP solver. Therefore, the potential of the proposed approach for real-time implementation is quite significant.

Further work can investigate the use of alternative networks to map states to Q-values such as convolutional or recurrent neural networks. Additionally, other studies can benchmark other reinforcement learning methods based on policy optimization to solve the problem, as opposed to value-based optimization methods.

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.

Acknowledgment

The authors would like to acknowledge the support provided by King Fahd University of Petroleum & Minerals through IRC-REPS funded project # INRE2103. The authors would like also to acknowledge the funding support provided by KACARE Energy Research and Innovative Center (ERIC), KFUPM.

References

[1] R.H. Lasseter, Smart distribution: Coupled microgrids, Proc. IEEE 99 (2011) 1074–1082.
[2] M.F. Zia, E. Elbouchikhi, M. Benbouzid, Microgrids energy management systems: A critical review on methods, solutions, and prospects, Appl. Energy 222 (2018) 1033–1055.
[3] J. E. Stamp, "Microgrid Modeling to Support the Design Process.," 2012.
[4] Renewable Capacity Statistics 2021," The International Renewable Energy Agency (IRENA), 2021. [Online]. Available: https://irena.org/publications/2021/March/Renewable-Capacity-Statistics-2021. [Accessed 27 5 2021].
[5] W. Su, J. Wang, J. Roh, Stochastic energy scheduling in microgrids with intermittent renewable energy resources, IEEE Trans. Smart Grid 5 (2013) 1876–1883.
[6] Z. Wang, B. Chen, J. Wang, M.M. Begovic, C. Chen, Coordinated energy management of networked microgrids in distribution systems, IEEE Trans. Smart Grid 6 (1) (2015) 45–53.
[7] F. Farzan, M.A. Jafari, R. Masiello, Y. Lu, Toward optimal day-ahead scheduling and operation control of microgrids under uncertainty, IEEE Trans. Smart Grid 6 (2) (2015) 499–507.
[8] Y.u. Zhang, N. Gatsis, G.B. Giannakis, Robust energy management for microgrids with high-penetration renewables, IEEE Trans. Sustainable Energy 4 (4) (2013) 944–953.
[9] F. Valencia, D. Saez, J. Collado, F. Avila, A. Marquez, J.J. Espinosa, Robust energy management system based on interval fuzzy models, IEEE Trans. Control Syst. Technol. 24 (1) (2016) 140–157.
[10] X. Tan, Q. Li, H. Wang, Advances and trends of energy storage technology in microgrid, Int. J. Electr. Power Energy Syst. 44 (1) (2013) 179–191.
[11] M. Faisal, M.A. Hannan, P.J. Ker, A. Hussain, M.B. Mansor, F. Blaabjerg, Review of energy storage system technologies in microgrid applications: Issues and challenges, IEEE Access 6 (2018) 35143–35164.
[12] M.K. Kiptoo, M.E. Lotfy, O.B. Adewuyi, A. Conteh, A.M. Howlader, T. Senjyu, Integrated approach for optimal techno-economic planning for high renewable energy-based isolated microgrid considering cost of energy storage and demand response strategies, Energy Convers. Manage. 215 (2020) 112917.
[13] P. Zeng, H. Li, H. He, S. Li, Dynamic energy management of a microgrid using approximate dynamic programming and deep recurrent neural network learning, IEEE Trans. Smart Grid 10 (4) (2019) 4435–4445.
[14] M. Patterson, N.F. Macia, A.M. Kannan, Hybrid microgrid model based on solar photovoltaic battery fuel cell system for intermittent load applications, IEEE Trans. Energy Convers. 30 (1) (2015) 359–366.
[15] K. Thirugnanam, S.K. Kerk, C. Yuen, N. Liu, M. Zhang, Energy management for renewable microgrid in reducing diesel generators usage with multiple types of battery, IEEE Trans. Ind. Electron. 65 (2018) 6772–6786.
[16] R. Palma-Behnke, C. Benavides, F. Lanas, B. Severino, L. Reyes, J. Llanos, D. Saez, A microgrid energy management

system based on the rolling horizon strategy, IEEE Trans. Smart Grid 4 (2) (2013) 996–1006.
[17] T. Morstyn, B. Hredzak, R.P. Aguilera, V.G. Agelidis, Model predictive control for distributed microgrid battery energy storage systems, IEEE Trans. Control Syst. Technol. 26 (2017) 1107–1114.
[18] N. Bazmohammadi, A. Anvari-Moghaddam, A. Tahsiri, A. Madary, J. Vasquez, J. Guerrero, Stochastic predictive energy management of multi-microgrid systems, Applied Sciences 10 (14) (2020) 4833, https://doi.org/10.3390/app10144833.
[19] N. Bazmohammadi, A. Tahsiri, A. Anvari-Moghaddam, J.M. Guerrero, A hierarchical energy management strategy for interconnected microgrids considering uncertainty, Int. J. Electr. Power Energy Syst. 109 (2019) 597–608.
[20] Y. Ye, D. Qiu, X. Wu, G. Strbac, J. Ward, Model-free real-time autonomous control for a residential multi-energy system using deep reinforcement learning, IEEE Trans. Smart Grid 11 (2020) 3068–3082.
[21] L. Yu, S. Qin, M. Zhang, C. Shen, T. Jiang and X. Guan, "Deep Reinforcement Learning for Smart Building Energy Management: A Survey," *arXiv preprint arXiv:2008.05074,* 2020.
[22] Y. Ji, J. Wang, J. Xu, X. Fang, H. Zhang, Real-Time Energy Management of a Microgrid Using Deep Reinforcement Learning, Energies 12 (12) (2019) 2291, https://doi.org/10.3390/en12122291.
[23] B. Mbuwir, F. Ruelens, F. Spiessens, G. Deconinck, Battery energy management in a microgrid using batch reinforcement learning, Energies 10 (11) (2017) 1846, https://doi.org/10.3390/en10111846.
[24] G.K. Venayagamoorthy, R.K. Sharma, P.K. Gautam, A. Ahmadi, Dynamic energy management system for a smart microgrid, IEEE Trans. Neural Networks Learn. Syst. 27 (2016) 1643–1656.
[25] E. Foruzan, L.-K. Soh, S. Asgarpoor, Reinforcement learning approach for optimal distributed energy management in a microgrid, IEEE Trans. Power Syst. 33 (2018) 5749–5758.
[26] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Ostrovski and others, "Human-level control through deep reinforcement learning,", Nature 518 (7540) (2015) 529–533.
[27] Y. Ye, D. Qiu, M. Sun, D. Papadaskalopoulos, G. Strbac, Deep reinforcement learning for strategic bidding in electricity markets, IEEE Trans. Smart Grid 11 (2020) 1343–1355.
[28] Q. Zhang, K. Dehghanpour, Z. Wang, Q. Huang, A learning-based power management method for networked microgrids under incomplete information, IEEE Trans. Smart Grid 11 (2020) 1193–1204.
[29] Y. Du, F. Li, Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning, IEEE Trans. Smart Grid 11 (2020) 1066–1076.
[30] Q. Huang, R. Huang, W. Hao, J. Tan, R. Fan, Z. Huang, Adaptive power system emergency control using deep reinforcement learning, IEEE Trans. Smart Grid 11 (2020) 1171–1182.
[31] R. Diao, Z. Wang, D. Shi, Q. Chang, J. Duan, X. Zhang, Autonomous voltage control for grid operation using deep reinforcement learning, 2020 IEEE Power & Energy Society General Meeting (PESGM) (2020).
[32] V. François-Lavet, D. Taralla, D. Ernst and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *European Workshop on Reinforcement Learning (EWRL 2016)*, 2016.
[33] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[34] V. François-Lavet, P. Henderson, R. Islam, M.G. Bellemare, J. Pineau, Pineau and others, "An introduction to deep reinforcement learning," *Foundations and Trends®*, Machine Learning 11 (3-4) (2018) 219–354.
[35] M. Riedmiller, Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method, European Conference on Machine Learning (2005).
[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602,* 2013.
[37] T. Schaul, J. Quan, I. Antonoglou and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952,* 2015.
[38] H. Van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv preprint arXiv:1509.06461,* 2015.
[39] H.S.V.S.K. Nunna, S. Doolla, Energy management in microgrids using demand response and distributed storage_A multiagent approach, IEEE Trans. Power Delivery 28 (2013) 939–947.
[40] Q. Jiang, M. Xue, G. Geng, Energy management of microgrid in grid-connected and stand-alone modes, IEEE Trans. Power Syst. 28 (2013) 3380–3389.
[41] A.A. Khan, M. Naeem, M. Iqbal, S. Qaisar, A. Anpalagan, A compendium of optimization objectives, constraints, tools and algorithms for energy management in microgrids, Renew. Sustain. Energy Rev. 58 (2016) 1664–1683.
[42] K. Strunz, E. Abbasi, C. Abbey, C. Andrieu, F. Gao, T. Gaunt, A. Gole, N. Hatziargyriou, R. Iravani, Benchmark systems for network integration of renewable and distributed energy resources, Cigre Task Force C 6 (2009) 78.
[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym (2016).
[44] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, M. Braun, Pandapower—An open-source python tool for convenient modeling, analysis, and optimization of electric power systems, IEEE Trans. Power Syst. 33 (2018) 6510–6521.
[45] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor and Y. Wu, *Stable Baselines,* GitHub, 2018.