Sustainable Energy, Grids and Networks 25 (2021) 100413

Contents lists available at ScienceDirect

Sustainable Energy, Grids and Networks

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

Deep reinforcement learning for energy management in a microgrid with flexible demand

Taha Abdelhalim Nakabi *, Pekka Toivanen

University of Eastern Finland, School of Computing, Kuopio Campus, P.O. Box 1627, 70211 Kuopio, Finland

ARTICLE INFO

Article history:
Received 14 February 2020
Received in revised form 9 November 2020
Accepted 20 November 2020
Available online 27 November 2020

Keywords:
Artificial intelligence
Deep reinforcement learning
Demand Response
Dynamic pricing
Energy management system
Microgrid
Neural networks
Price-responsive loads
Smart grid
Thermostatically controlled loads

ABSTRACT

In this paper, we study the performance of various deep reinforcement learning algorithms to enhance the energy management system of a microgrid. We propose a novel microgrid model that consists of a wind turbine generator, an energy storage system, a set of thermostatically controlled loads, a set of price-responsive loads, and a connection to the main grid. The proposed energy management system is designed to coordinate among the different flexible sources by defining the priority resources, direct demand control signals, and electricity prices. Seven deep reinforcement learning algorithms were implemented and are empirically compared in this paper. The numerical results show that the deep reinforcement learning algorithms differ widely in their ability to converge to optimal policies. By adding an experience replay and a semi-deterministic training phase to the well-known asynchronous advantage actor–critic algorithm, we achieved the highest model performance as well as convergence to near-optimal policies.

1. Introduction

The ongoing transformation of the power network is mainly related to the transition from conventional centralized energy resources to distributed energy resources (DERs) that have low impacts on the environment. This transition requires innovative solutions to deal with the challenges arising from the intermittent nature of renewable energy resources. Smart grid technologies such as advanced metering infrastructures, energy storage systems (ESSs), and home energy management systems are being deployed around the globe to support this transformation. Microgrid systems use these technologies alongside DERs to efficiently meet the local power demand and support the decentralization of the power supply. Microgrids are usually low-voltage networks with a limited local electricity supply and demand compared with the main grid. Microgrids can either operate in parallel with the grid, buying and selling energy through the electricity market, or autonomously, using local generation and storage [1]. Therefore, they offer technical and economic benefits, including system reliability, local energy delivery, and additional sources of capital investment for the DERs.

We are interested in the higher-level control that consists of an energy management system (EMS) that maintains the energy reserve, maximizing the overall system efficiency and optimizing the dispatch of local resources. Because of the nature of the microgrid, the EMS faces major challenges, primarily related to the small scale, volatility, uncertainty, and intermittency of DERs, as well as the demand uncertainty and the dynamic electricity market prices. To overcome these challenges, further improvements in microgrid architecture and control are required. On the architecture level, additional sources of flexibility must be exploited to balance the high volatility of DERs. In addition, new control mechanisms and intelligent control methods are needed to optimize the energy dispatch and overcome the uncertainties of the microgrid components.

Typically, microgrid components include DERs, electric loads, and an ESS. The DERs consist of renewable energy resources, typically based on wind turbines [2] or solar PV [3], and commonly backed up by an energy generator using a natural gas [4] or diesel engine [5]. The emerging interest in DERs stems from the many advantages that they can offer. DERs' small cost and their short construction lead times compared to larger central power plants put them in advantage in growing and liberalized electricity markets. DERs can in some cases eliminate the need of expanding the grid which reduces the costs of building and maintaining high voltage power lines. Additionally, using DERs reduces the need of central dispatch and the energy losses in long distance transmissions. Finally, DERs can provide the microgrid with resiliency as it will keep the local supply in case of disturbances in the grid. The electric load components of a microgrid

* Corresponding author.
E-mail addresses: tahanak@uef.fi (T.A. Nakabi), Pekka.Toivanen@uef.fi (P. Toivanen).

https://doi.org/10.1016/j.segan.2020.100413
2352-4677/

can be either residential loads [6] or industrial loads [7]. The ESSs are typically based on batteries and can either be distributed in the microgrid [8] or centralized [9].

Several innovative components have been presented in the literature to improve the reliability and flexibility of microgrids. Innovative electric loads, which can offer high demand-side flexibility, have been proposed in several works. These include directly controllable loads [10], thermostatically controlled loads (TCLs) [6,11,12], price-responsive loads [11,13], and electric vehicles [7,14]. However, the combination of these demand-side flexibility sources in a microgrid has been rarely studied in the literature. In this paper, we propose to increase the flexibility in demand by combining groups of TCLs and price-responsive loads participating in a demand response (DR) program, alongside a shared ESS, a wind power resource, and a connection to the main grid. TCLs can provide significant flexibility due to their thermal conservation of energy [15,16], whereas price-responsive loads can offer more flexibility by shifting their consumption to periods of high energy production [17].

On the control level, we differentiate between two categories of EMS, namely, model-based EMS and model-free EMS. In model-based approaches, an explicit model is used to formulate the dynamics of the microgrid and the different interactions between its components. The uncertainties are estimated using a predictor using deterministic or probabilistic forecast models based on historical data and estimated parameters. Whereas the control problem is solved using a scheduling optimizer. Model predictive control is the most commonly and successfully used algorithm in the literature [18–20], consisting of repeated optimizations of the predictive model over a progressing time period. Model-based approaches rely heavily on domain expertise for constructing accurate models and parameters for a microgrid. Therefore, model-based approaches are not transferable nor scalable, which leads to high development costs. Furthermore, if the uncertainties in the microgrid change over time, the model, predictor, and solver must be redesigned correspondingly, which significantly increases the maintenance costs.

Model-free or data-driven approaches consist of learning abstract representations of near-optimal control strategies in the microgrid from its operational data. Learning-based methods have been introduced in recent years as an alternative to model-based approaches, as they can reduce the need for an explicit system model, improve the EMS scalability, and reduce the maintenance costs of the EMS [21]. One of the most promising learning-based EMS methods is the reinforcement learning (RL) paradigm [22], in which an agent learns the dynamics of the microgrid by interacting with its components. Several works have demonstrated successful implementations of reinforcement learning-based EMSs in different microgrid architectures, either within a single agent [23–25] or multi-agent framework [26–29]. However, the basic and most popular RL methods, such as Q-learning [30], face several challenges related to inefficient data usage, high dimensionality, state space continuity, and transition function uncertainty. A batch RL algorithm was proposed in [23] to overcome the problem of inefficient data usage using a batch of past experiences. In [28], a fuzzy Q-learning algorithm was proposed to cope with the continuous nature of the state and action spaces. In [31], an extremely randomized trees algorithm was proposed as a regression algorithm for state–action pairs, to solve the problem of uncertainty. Deep reinforcement learning (DRL) methods, however, use artificial neural networks (ANN) as function approximators, capable of learning continuous state–action transitions under uncertainty [32]. The neural networks enable the use of continuous and high-dimensional state spaces and can extract hidden features from the state space. This enables the DRL agent to overcome the uncertainty and partial observability of the environment [33].

Driven by the recent successes of DRL in solving complex tasks, such as the superhuman performance achieved using AlphaZero in many challenging games [34], several works have shown interest in DRL applications for microgrid control problems. Based on the control mechanisms, specifically the effect of the control actions on the components of the microgrid, two approaches to DRL-based EMSs can be distinguished in the literature. The first approach consists of simple control mechanisms that manage individual components of the microgrid. Optimal management of the ESS under uncertainties in electricity consumption and production has been proposed using a variety of methods, including deep Q-learning (DQN) [35], SARSA [36], and double DQN [37]. In [38], a batch DQN was proposed to optimally control a cluster of TCLs. The second approach revealed in the literature consists of jointly managing multiple components of the microgrid using complex objective functions. In this approach, the DRL algorithm is based on a complex action space, which combines the actions related to each component. DQN algorithms have been used to optimize local energy trading and sharing by either controlling battery charging/discharging and buying/selling operations with the main grid [39] or managing the consumer's energy sharing options [40]. Another energy exchange strategy, based on the energy internet concept, was proposed in [41], in which the control mechanisms included an A3C algorithm for management of the backup generators, fuel cells, and ESS. A real-time energy management approach using a DQN algorithm was proposed in [42] to jointly schedule backup generator utilization and manage ESS and grid operations, whereas [43] proposed jointly controlling the hydrogen storage, diesel generation, and ESS using the same algorithm. However, the control mechanisms adopted in these works did not consider DR programs as a flexibility provider in the microgrid. DR acts as an indirect control mechanism, which provides optimal incentives, or price signals that steer the consumption toward periods of high production.

Furthermore, several works have proposed RL methods to control the consumption and exploit demand flexibility in smart grids [44–46]. However, the combination of DR programs and direct control mechanisms, such as TCL control, in the EMS of a microgrid is still absent from the literature. Therefore, we propose a novel microgrid EMS architecture that makes optimal use of the flexibility present, given exogenous conditions such as demand, external network prices and the wind energy resources. The methodology combines between direct control of typical microgrid components, direct control of a TCL cluster, and indirect control of price-responsive loads through price-based DR. Moreover, the abovementioned DRL methods focused mainly on DQN algorithms and rarely investigated the recent policy gradient and actor–critic algorithms. Additionally, no thorough comparison of the algorithm performances has been reported in the context of a microgrid EMS. Therefore, in this paper, we present and compare the performances of seven state-of-the-art DRL algorithms and two baseline control methods. We also propose two improvements on the A3C and Proximal policy optimization (PPO) methods, which demonstrate better performances than the other algorithms proposed in this study. The algorithms are tested in different scenarios using a realistic microgrid simulation, based on real electricity price and renewable energy production data from Finland. The performance of each algorithm is evaluated through the lenses of gross energy profit from operations, and optimal use of local resources and flexibility components. Other issues addressed in this paper include DRL algorithm overfitting and the premature convergence to suboptimal deterministic policies.

The major contributions of this study are as follows:

- A novel microgrid model that includes TCLs and price-responsive loads as flexibility resources alongside the typical microgrid components.

- A novel control mechanism that combines direct control of TCLs with indirect control of price-responsive loads, alongside priority management of the ESS and main grid in case of energy deficiency or excess.
- A Markov-Decision process (MDP) formulation of the control problem, considering the various control actions, along with a reward function based on the gross energy profit i.e. the generated revenue from selling electricity locally and to the grid, minus the cost of energy generated and imported from the grid.
- A comprehensive and numerical comparison of value-based DRL algorithms (DQN, Double DQN, SARSA) and policy-based algorithms (REINFORCE, Actor–critic, A3C, PPO).
- Novel variations of the A3C and PPO algorithms, which incorporate an additional experience replay to avoid inefficient data usage and destructive searching, and a semi-deterministic training to exploit the optimal local policies. The proposed variations outperformed the abovementioned methods.

The remainder of this paper is organized as follows: In Section 2, we present the microgrid model and the EMS control mechanisms. In Section 3, we formulate the problem as a Markov decision process. Section 4 presents a theoretical framework for the DRL algorithms used in this study. Experimental implementation details are presented in Section 5. Numerical results for each method and their numerical comparisons are presented and discussed in Section 6. Section 7 presents our conclusions.

2. Microgrid model and problem formulation

This study considers a microgrid with an independent supply and demand infrastructure. The microgrid is managed by an aggregator or a utility company that is responsible for supplying the electricity to meet the local demand. The microgrid has its own wind-turbine-based DER but is also connected to the main grid, through which it continuously buys or sells energy on the electricity markets.

The architecture of the microgrid is illustrated in Fig. 1. It consists of three layers: the physical, information, and control layers. The physical layer includes a wind-based DER, a communal ESS, a group of TCLs, and a group of residential price-responsive loads. The information layer consists of the smart meters and the system for two-way communication between each of the individual components and the EMS. Information such as electricity prices, battery states of charge, and energy generation is transferred through this layer. The control layer represents the infrastructure through which the EMS sends control signals to the controllable components of the grid. As illustrated in Fig. 1, there are three direct control points, namely, the TCL on/off control, ESS charge/discharge control, and energy grid buy/sell control. Given this architecture, we modeled our microgrid as a multi-agent system in which each component operated as an autonomous agent, interacting with the environment, and other agents. The simple or complex behavior of each component was governed by an internal model. In the following sections, we present the models adopted for each component of the microgrid.

2.1. Energy storage system model (ESS)

For technical and economic convenience [47], we adopted a community ESS instead of individual household battery storages. The utilized ESS is capable of covering a minimum of 2 h of the energy demand of the microgrid. At each time step t, the storage dynamics of the ESS were modeled by

$$B_t = B_{t-1} + \eta_c C_t - \frac{D_t}{\eta_d}, \tag{1}$$

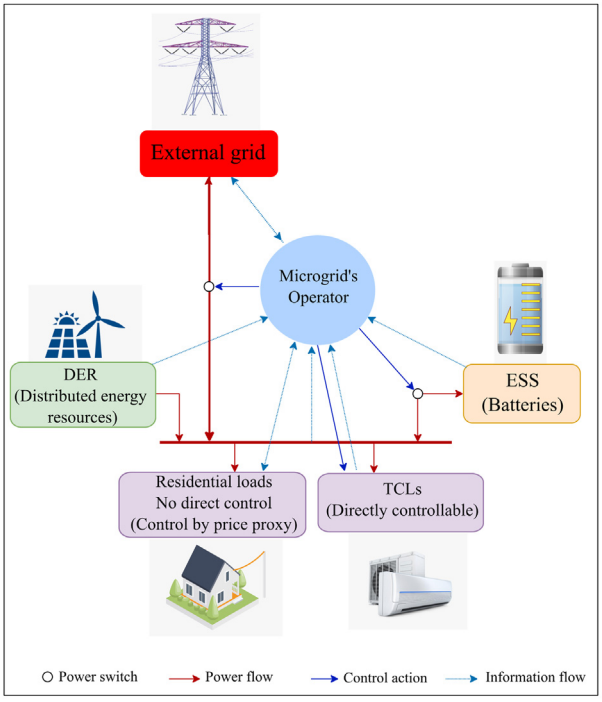

Fig. 1. The proposed microgrid architecture.

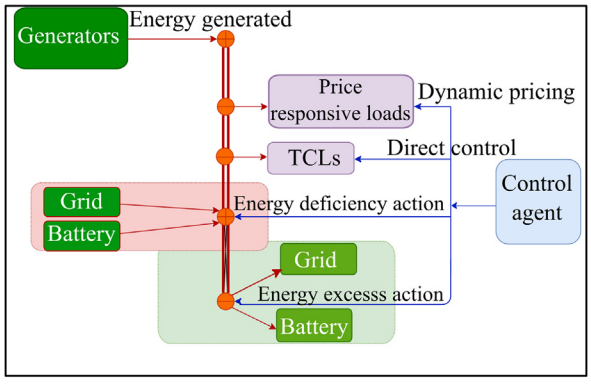

Fig. 2. The control mechanisms used in the microgrid management system.

where $B_t \in [0, B_{\max}]$ is the stored energy in the ESS at time t, $B_{\max}$ is the ESS maximum capacity, and $(\eta_c, \eta_d) \in]0, 1]^2$ are the charging and discharging efficiency coefficients, respectively. The variables $C_t \in [0, C_{max}]$ and $D_t \in [0, D_{max}]$ are the charging and discharging powers, respectively, which are constrained by the ESS charging and discharging rate limitations C_{max} and D_{max}. We also defined the ESS state-of-charge variable as

$$BSC_t = \frac{B_t}{B_{\max}}. \tag{2}$$

The behavior of the ESS in response to the charge/discharge control signals is represented by the energy provided to and requested from the batteries. In the case of a charging signal from the EMS, the ESS agent receives a power rate for storage in the batteries, verifies the feasibility of the charging operations (based on maximum capacity and maximum charging rate), stores the energy accordingly, and returns the remaining power to be sold to the main grid. Similarly, in the discharging case, the ESS agent receives a power request from the EMS, verifies the supply conditions, and returns the available power accordingly. If the

requested power cannot be completely supplied by the ESS, the difference is automatically supplied from the main grid.

2.2. Distributed energy resource model (DER)

The microgrid in this study is considered to be equipped with wind turbines capable of generating varying energy quantities, depending on the weather conditions. Instead of using a model for the energy generation, we utilized real wind energy production data from a wind farm in Finland [48]. The DER sensor shares the information about the current energy generation G_t, with the EMS, and supplies the energy generated directly to the local grid.

2.3. Main electricity grid

The microgrid is connected to a main grid that acts as a regulation reserve. The supply and demand in the microgrid cannot be balanced using the DERs alone, due to the intermittent, and uncontrollable nature of these resources. The main electricity grid can instantly supply power to the microgrid in the case of an energy deficiency or accept the excess power in the case of surplus. The transactions between the main grid and the microgrid happen in real-time using the up-regulation and down-regulation market prices. The main grid agent shares the real-time up- and down-regulation prices, represented respectively as $\left(P_t^u, P_t^d\right)$, with the EMS. In the model, we implemented real up- and down-regulation price data from the balancing electricity market in Finland provided in [49]. To define the priority supply source in the case of a deficiency and the priority power discharge source in the case of an excess, the EMS controls only the electrical switch to the main grid, as illustrated in Fig. 2. After each time step, the EMS receives information on the energy E_t, purchased, or sold to the main grid, where positive values indicate purchased energy and negative values sold energy.

2.4. Thermostatically controlled loads (TCLs)

A cluster of TCLs can provide a significant source of flexibility because of their thermal conservation of energy. We assumed that most of the households in the microgrid were equipped with a TCL, such as an air conditioner, heat pump, water heater, or refrigerator. These TCLs are directly controllable at each time step t, using control signals from the TCL aggregator. We propose that, for fairness to the TCL consumers, the access to the direct control of TCLs is compensated by a lower price for the electricity consumed by these loads. Therefore, the TCLs will only be charged the cost of power generation C_{gen}. In order to preserve the end user's comfort levels, each TCL is equipped with a backup controller that maintains the temperatures in an acceptable range. The backup controller receives the on/off action u_t^i, from the TCL aggregator, verifies the temperature constraints, and modifies the action as follows:

$$u_{b,t}^i = \begin{cases} 0 & if \quad T_t^i > T_{max}^i \\ u_t^i & if \quad T_{min}^i < T_t^i < T_{max}^i \\ 1 & if \quad T_t^i < T_{min}^i, \end{cases} \tag{3}$$

where $u_{b,t}^i$ is the final on/off action after the decision of the backup controller; T_t^i is the operational temperature of TCL i at time t; and T_{max}^i and T_{min}^i are the upper and lower temperature boundaries set by the end user, respectively. The temperature dynamics of each TCL were modeled using a second-order model based on [38]

$$\begin{aligned} \dot{T}_t^i &= \frac{1}{C_a^i}\left(T_t^0 - T_t^i\right) + \frac{1}{C_m^i}\left(T_{m,t}^i - T_t^i\right) + L_{\mathrm{TCL}}^i u_{b,t}^i + q^i \\ \dot{T}_{m,t}^i &= \frac{1}{C_m^i}\left(T_t^i - T_{m,t}^i\right) \end{aligned} \tag{4}$$

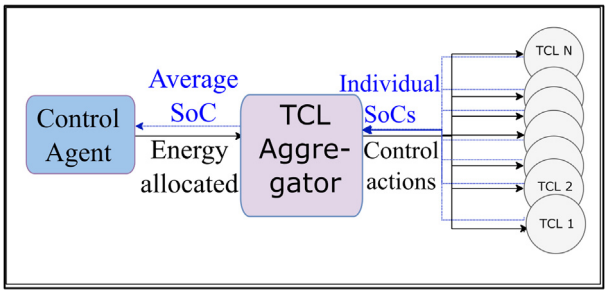

Fig. 3. The intermediary role of the TCL aggregator.

where T_t^i is the measured indoor air temperature; $T_{m,t}^i$ is the non-observable building mass temperature; T_t^0 is the outdoor temperature; C_a^i and C_m^i are the thermal masses of the air and the building materials, respectively; q^i is the internal heating in the building; and L_{tcl}^i is the nominal power of the TCL. Finally, a state-of-charge measure SoC_t^i, which determines the relative position of T_t^i in the desired temperature range, was defined for each TCL as

$$\mathrm{SoC_t^i} = \frac{T_t^i - T_{min}^i}{T_{max}^i - T_{min}^i}. \tag{5}$$

2.5. Residential price-responsive loads

The residential loads represent the electricity demand from households in the microgrid that cannot be directly controlled. We assumed that these loads follow a daily pattern, with a variable component that can be affected by the electricity prices. In this model, each household i is characterized by 2 parameters, a sensitivity factor $\beta_i \in]0, 1]$ and a patience parameter λ_i. The sensitivity is the percentage of load that can be increased or decreased given respectively a decrease or an increase in the price. The patience is the number of hours, during which the shifted loads are paid back. The electric load L_t^i of household i at time t is modeled using the following equations:

$$L_t^i = L_{b,t} - \mathrm{SL_t^i} + \mathrm{PB_t^i} \tag{6}$$

$$SL_t^i = L_{b,t} * \beta_i * \delta_t \tag{7}$$

where $L_{b,t} > 0$ and indicates the basic load, which follows the daily consumption pattern [50]. This pattern can be inferred from the average daily consumption curve of the residential area in which the microgrid is implemented. $\mathrm{SL_t^i}$ is the shifted load defined by (7), where $\delta_t \epsilon \{-2, -1, 0, 1, 2\}$ is the price level at time t. Therefore, SL_t^i is positive for high price levels $\delta_t > 0$ and negative for low price levels $\delta_t < 0$. $\mathrm{PB_t^i}$ are the loads shifted from previous timesteps to be paid back. The positive shifted loads from a certain hour are ought to be executed after a certain number of hours, and the negative shifted loads will be withheld from future time steps as they were executed in advance. These payback quantities are represented by:

$$PB_t^i = \sum_{j=0}^{t-1} \omega_{i,j} * \mathrm{SL_j^i} \tag{8}$$

where $\omega_{i,j} \epsilon \{0, 1\}$ represents the execution decision of the load shifted from time step j. $\omega_{i,j}$ is determined by a stochastic variable that depends on the current price level δ_t, and the time passed since the timestep j. The closer this period gets from the maximum patience λ_i, the higher is the probability of the load being executed ($\omega_j = 1$). We formulate this mechanism by the

following probability:

$$P\left(\omega_{i,j}=1\right)=clip\left(\frac{-\delta_t * sign\left(\mathrm{SL}_j^i\right)}{2}+\frac{(t-j)}{\lambda_i},0,1\right) \tag{9}$$

$$clip\left(X,a,b\right)=\begin{cases} a\ if\ X<a \\ X\ if\ a\le X\le b \\ b\ if\ X>y \end{cases}. \tag{10}$$

The sign(SL_j^i) controls the probability in a way that the positive shifted loads from past timesteps are less likely to be executed if the price is high, and vice-versa for the negative shifted loads.

This represents a notably simplistic model of the price-responsive loads. However, this model is only used herein as a proof of concept for DRL methods that can learn an abstract representation of the residential load price responsiveness based only on the reward feedback.

2.6. EMS agent

The proposed EMS agent uses the information provided by the different grid components and the observable environment to determine the optimal supply/demand balancing strategy. The agent performs overall management of the microgrid using four control mechanisms: TCL direct control, price level control, energy deficiency actions, and energy excess actions. These mechanisms are illustrated in Fig. 2 and detailed in the following sections.

2.6.1. TCL direct control

At each time step t, the EMS agent allocates a certain amount of energy for use in TCL operations. This energy is then dispatched through an intermediate agent, the TCL aggregator, to the individual TCLs. Based on the energy allocation issued by the EMS agent, the aggregator determines the on/off actions of each TCL based on their SoC priority; TCLs with the lowest SoC values are served before TCLs with higher SoCs. The TCL aggregator also operates as an information aggregator, by communicating the real-time average SoC of the TCL cluster to the EMS agent, as illustrated in Fig. 3. It is worth noting that because of the backup controller at each TCL, the allocated energy is not always equal to the actual energy used by the TCLs.

2.6.2. Pricing mechanism

The EMS agent determines the price level δ_t to use at every time step in order to exploit the households' price elasticity. It is worth mentioning that the microgrid manager is not the system operator and does not have any monopoly over the demand. To insure the competitivity of the microgrid's manager business model, we designed the pricing mechanism in a way that the prices offered can fluctuate around an median value while the daily price average P_{avg} does not exceed the market price offered by electricity retailers P_{market} by more than 2.9% . For practical reasons related to the action space's discontinuity discussed in the next section, we consider that the DR program is based on discrete price levels [51], the fluctuation is controlled by the price level δ_t and the price can only take one of 5 values:

$$P_t \in \left(P_{market}+\delta_t * cst\right)_{\delta_t \epsilon \{-2,-1,0,1,2\}} \tag{11}$$

with cst a constant determining the increment or decrement in the price. The variations of the price serve as a tool for shifting the loads from peak periods toward periods with more power availability. However, the agent cannot raise the prices for long periods of the day. Hence, we implemented a systematic constraint on the price to keep $\frac{P_{avg}-P_{market}}{P_{market}} < 2.9\%$. We keep record of the price levels applied at each time step and whenever the sum of the price levels from previous timesteps is higher than a threshold, the price offered is P_{market} instead of the price proposed by the agent. Therefore, the effective price level $\delta_{t,eff}$ is defined by:

$$\delta_{t,eff}=\begin{cases} \delta_t\ if\ \sum_{j=0}^{t}\delta_t \le threshold \\ 0\ if\ \sum_{j=0}^{t}\delta_t > threshold \end{cases} \tag{12}$$

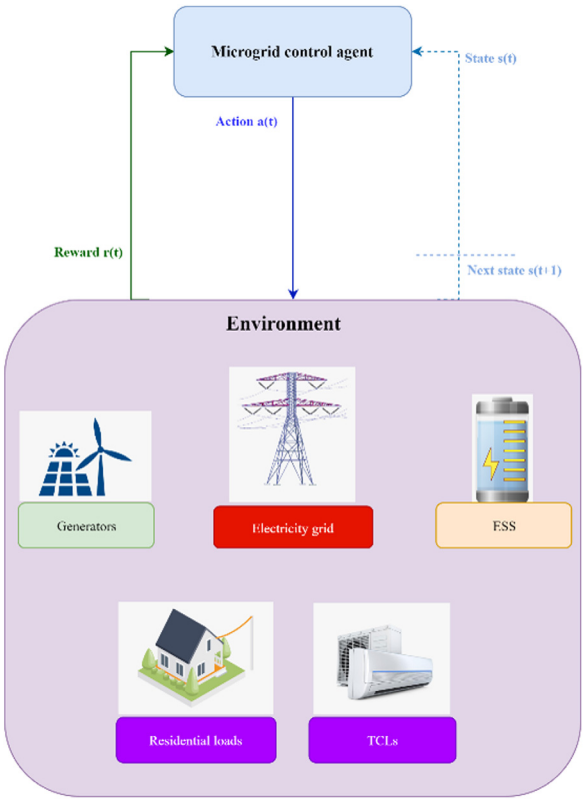

Fig. 4. Interaction process between the control agent and microgrid environment.

2.6.3. Energy deficiency action

When the local DERs are not able to meet the demand, the local microgrid can either use the energy stored in the ESS or purchase energy from the main grid and save the ESS energy for later use. At each time step, the EMS agent sets the usage priority among these two resources. The microgrid is AC and it is synchronized with the grid. Therefore, if the priority resource is the ESS and the required energy cannot be entirely fulfilled, the remaining demand is automatically supplied from the main grid.

2.6.4. Energy excess action

The energy generated by the local DERs can also exceed the demand. In this case, the excess energy must be either stored in the ESS or sold to the main grid. The EMS agent specifies the priority option for excess energy usage in advance, similarly to in the energy deficiency scenario. If the ESS is the priority option and the battery capacity is reached, the remaining energy is automatically transferred to the main grid.

3. Markov decision process formalism

The RL paradigm refers to the set of control methods in which an agent learns the optimal control policies by interacting with

an environment. In this study, this agent learning was achieved through a Markov decision process formalism [52]. At each time step, the agent performs an action based on the current state of the environment and, in return, receives a reward, and information about the next state, as illustrated in Fig. 4. In order to find the optimal policy, the agent must estimate the quality of its actions using information from the previously explored states of the environment. In the following MDP formulation, the MDP is characterized by a state space S, an action space A, a transition function T, and a reward function R. The transition function T, describes the probability of a transition from state $s \in S$ to $s' \in S$ given an action a, such that

$$T: S \times A \times S \rightarrow [0, 1],$$
$$T\left(s, a, s'\right) = p\left(s_{t+1} = s' | s_t = s, a_t = a\right). \tag{13}$$

The reward function $R \in \mathbb{R}$ describes the immediate reward received by the agent after it performs the transition to state s' from state s, given action a.

The objective is to optimize a stochastic policy π, which is a distribution over actions given states, mathematically represented

$$\pi: S \rightarrow [0, 1]^{N_A},$$
$$\pi\left(s\right) = \left(p\left(a|s\right)\right)_{a \in A}. \tag{14}$$

The notation $\pi\left(a \parallel s\right) = p\left(a|s\right)$ will be used to refer to the probability of choosing action a given a state s. At each time step t, the agent receives a state s_t and selects an action a from the set of possible actions A, according to its policy π. In return, the agent receives the next state s_{t+1} and a reward R_t. The process continues until the agent reaches a terminal state, after which the process restarts. Following policy π, the goal is to maximize the cumulative discounted reward, given by:

$$\rho_t = \sum_{k=t}^{D-1} \gamma^{k-t} R_k, \tag{15}$$

where $\gamma \in]0, 1]$ is the discount factor, which determines the importance of the rewards in the next step compared with that of the immediate reward. The state value under policy π is simply the expected return for following policy π from state s:

$$V^{\pi}: S \rightarrow \mathbb{R},$$
$$V^{\pi}\left(s_t\right) = E_{\pi}\left[\rho_t\right]. \tag{16}$$

V^{π} can also be defined as

$$V^{\pi}\left(s_t\right) = E_{\pi}\left[R_t + \gamma V^{\pi}\left(s_{t+1}\right)\right], \forall t < D - 1, \tag{17}$$

At the last time step D − 1:

$$V^{\pi}\left(s_{D-1}\right) = E_{\pi}\left[R_{D-1}\right]. \tag{18}$$

Conversely, the action-value function, or Q-function, describes the expected return for selecting an action a_t, at state s_t, and following policy π onward:

$$Q^{\pi}: S \times A \rightarrow \mathbb{R},$$
$$Q^{\pi}\left(s_t, a_t\right) = R_t + \gamma V^{\pi}\left(s_{t+1}\right). \tag{19}$$

The agent begins searching for the optimal policy π^*, at an initial state s_0, that maximizes the action-value function:

$$\pi^* = \underset{\pi}{arg\,max}\, Q^{\pi}\left(s, a\right), \tag{20}$$
$$Q^*\left(s, a\right) = Q^{\pi^*}\left(s, a\right). \tag{21}$$

The following section describes the states, actions, and reward function specific to our problem formulation.

3.1. State description

Based on the problem formulation in the previous section, the state space consists of the information that the agent uses in the decision-making process at each time step t. The state consists of a controllable state component S^C, an exogenous state component S^X, and a time-dependent component S^T. The controllable state information includes all environmental variables that the agent affects directly or indirectly. In this work, the average SoC of the TCLs, the state-of-charge of the ESS BSC_t, and the pricing counter C_t^b comprise the controllable state. The exogenous information consists of all the variables that the agent has no control over, such as the temperature T_t, energy generation G_t, and electricity prices at the regulation market P_t^u. We assumed that the controller could accurately forecast these three variables for the following hour. The time-dependent component reflects the time-dependent behavioral patterns of the environment. In this work, S^T denotes the hour of day t, and $L_{b,t}$ the current load value of the daily consumption pattern. Therefore, the state space can be described as

$$\begin{aligned} s_t \in S &= S^C \times S^X \times S^T, \\ s_t &= \left[\mathrm{SoC_t}, BSC_t, C_t^b, T_t, G_t, P_t^u, L_{b,t}, t\right]. \end{aligned} \tag{22}$$

3.2. Action description

The agent acts on the environment using the control mechanisms described in Section 2. The action space consists of four components: the TCL action space A_{tcl}, price action space A_P, energy deficiency action space A_D, and energy excess action space A_E. The TCL action space consists of four possible actions that specify the energy allocation level for the TCLs, the price action space consists of five possible actions that specify the price level, and the energy deficiency, and energy excess action spaces each have two possible actions that specify the battery or grid priorities. Therefore, the action space represents the 80 potential combinations of the possible actions of these four components, given by

$$\begin{aligned} a_t &= \left(a_{tcl}, a_P, a_D, a_E\right)_t, \\ a_t \in A &= A_{tcl} \times A_P \times A_D \times A_E. \end{aligned} \tag{23}$$

3.3. Reward function

We designed our reward function to maximize the economic profit from operations. The reward is calculated as the gross margin from operations i.e. the revenues made by selling electricity to the microgrid and to the external grid, minus the costs related to power generation, purchases, and transmission from the external grid. Therefore, the reward function R_t is defined as:

$$R_t = Rev_t - Costs_t. \tag{24}$$

$$Rev_t = P_t \sum_{loads} L_t^i + C_{gen} \sum_{\mathrm{TCLs}} L_{\mathrm{TCL}}^i u_{b,t}^i + P_t^D E_t^S, \tag{25}$$

$$Costs_t = (P_t^u + C_{tr_{imp}}) E_t^P + \mathrm{C_{tr_{exp}}} E_t^S, \tag{26}$$

where P_t^D and P_t^{U} are the down-regulation and up-regulation prices, i.e. the prices at which the energy is sold to and bought from the external grid. G_t, E_t^S, and E_t^P are respectively the energies generated, sold to, and purchased from the external grid. C_{gen} is the power generation cost charged to TCLs, as mentioned before. $C_{tr_{imp}}$ and $C_{tr_{exp}}$ are the power transmission costs respectively for importing from and exporting to the external grid.

4. Proposed methods

In this section, we present an overview of the DRL methods used in this work. In most of the DRL algorithms presented in this study, the search follows an ε-greedy strategy. During the training process, at each training iteration k, the selection of an action follows a stochastic process by choosing a random action $a \in A$ with a probability ε_k:

$$\mathrm{p}\,(\mathrm{a} = \mathrm{random\_action}) = \varepsilon_{\mathrm{k}} \tag{27}$$

The ε-greedy strategy allows a combination between exploration and exploitation of the search space by starting at a high value of ε and decaying it throughout the training process.

$$\varepsilon_{\mathrm{k}} = \max(\varepsilon_{\mathrm{start}} - \varepsilon_{decay} * k, \varepsilon_{end}) \tag{28}$$

where $\varepsilon_{\mathrm{decay}}$ is the decay speed. A high value of ε_{decay} results in a fast convergence to a policy and less exploration, and vice-versa. After training, the actions' selection will directly follow the optimal policy π^*, i.e. $\varepsilon = 0$.

The DRL methods are divided into two main categories, namely, value-based, and policy-based methods. In value-based methods, the neural network learns the Q-function of each action given a state s, whereas in policy-based methods, the neural network learns a probability distribution of the actions given a state s.

$$DRL: \begin{cases} state - \rightarrow DNN - \rightarrow Q\,(s, a)\,, Value - based\,DRL \\ state - \rightarrow DNN - \rightarrow \pi\,(a \parallel s)\,, Policy - based\,DRL \end{cases}$$

4.1. Value-based DRL methods

Based on the MDP formulation notations, the Q-function is represented as an approximator using a neural network with parameters θ, as shown in Fig. 5. Deep Q-learning (DQN) is one of the most commonly used algorithms to tune the parameters θ and aims at directly approximating the optimal Q-function: $Q(s, a, \theta) \approx Q^*(s, a)$. The action selection is based directly on Q^* i.e:

$$a = arg\underset{a'}{max}\, Q^*\left(s, a'\right) \tag{29}$$

Below, we present the three DQN variations used in this work:

4.1.1. One-step DQN

In one-step DQN, the parameters θ, are learned by iteratively minimizing a sequence of loss functions, where the ith loss function is defined as

$$L_i(\theta_i) = E[(R_t + \gamma \underset{a'}{max}\, Q\left(s_{t+1}, a', \theta_{i-1}\right) - Q\,(s_t, a_t, \theta_i))^2] \tag{30}$$

The Q-function is updated toward the one-step return $R_t + \gamma max_{a'} Q\left(s', a', \theta_{i-1}\right)$. To increase the efficient usage of previously accumulated experience, we also included an experience replay mechanism [53]. In experience replay, the learning phase is logically separated from experience gain. Experience replay uses randomly sampled batches of transitions (s_t, a_t, R_t, s_{t+1}) from an experience dataset. Through this process, the neural network can overcome the limitation of non-stationary data distributions, resulting in better algorithm convergence. It is worth noting that this algorithm does not use the ε-greedy strategy as the exploration of the search space is always random during the training.

4.1.2. SARSA

The deep SARSA algorithm [54] is an on-policy version of DQN. Therefore, updates to the parameters θ are related to the actual agent behavior instead of the optimal actions. The SARSA loss function is defined as

$$L_i(\theta_i) = E\left[\,(R_t + \gamma Q\,(s_{t+1}, a_{t+1}, \theta_{i-1}) - Q\,(s_t, a_t, \theta_i))^2\right]. \tag{31}$$

The target value used by SARSA is $R_t + \gamma\,(s_{t+1}, a_{t+1}, \theta_{i-1})$, where a_{t+1} is the action taken in state s_{t+1}. SARSA learns a near-optimal policy while following an ε-greedy exploration strategy.

Algorithm 1: DQN with experience replay and target net

```
Initialize replay memory RM, with experiences collected by the random agent.
Initialize DQN network parameters θ randomly
Initialize target network parameters θ' ← θ
For episode = 1, Max_episodes do:
  For t=0, D-1 do:
    Get initial state s_t
    Take action a with ε-greedy policy-based on Q(s_t, a_t; θ)
    Receive new state s_{t+1} and reward, R_t
    Store transition (s_t, a_t, R_t, s_{t+1}) in RM
    Sample a random minibatch of transitions (s_t, a_t, R_t, s_{t+1})
    For each sample, set
      y_t = { R_t  if t = D − 1
            { R_t + γ max_{a'} Q(s_{t+1}, a', θ')  otherwise
    Perform a gradient descent step on: E[(y_t − Q(s_t, a_t, θ_i))^2] w.r.t to θ.
    Every TN steps, update the target network θ' ← θ.
  End for
  Reset environment
End for
```

4.1.3. Double DQN/target network

Double DQN [55] uses a separate neural network for target setting, called a *target network*. The target network is a frozen copy of the original network that has a lower update frequency for the parameters θ'. The loss function becomes

$$L_i(\theta_i) = E[(R_t + \gamma \underset{a'}{max}\, Q\left(s_{t+1}, a', \theta'\right) - Q\,(s_t, a_t, \theta_i))^2]. \tag{32}$$

After several steps, the target network is updated by copying the parameters from the original network:

$$\theta' \leftarrow \theta$$

To be effective, the interval between updates must be large enough to provide adequate time for the original network to converge. The target network provides stable Q-function targets for the loss function, which allows the original network to converge to the desired Q-function. Algorithm 1 outlines the pseudocode for the DQN with a target network. In practice, we first ran a random agent (taking random actions) to collect experiences from the environment and store the transitions (s_t, a_t, R_t, s_{t+1}) in the replay memory.

4.2. Policy-based DRL methods

In contrast to value-based methods, policy-based methods [56] aim to directly find an optimal policy π by parametrizing the

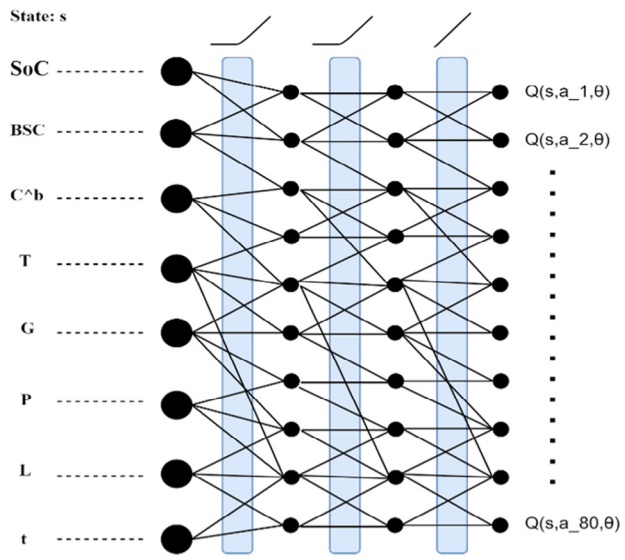

Fig. 5. The DQN neural network architecture.

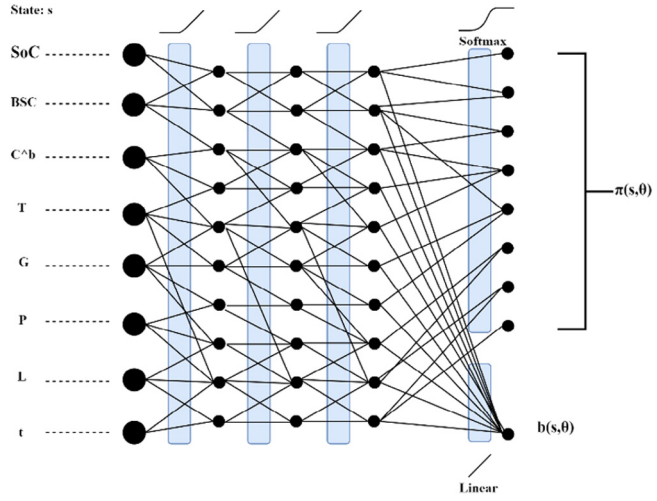

Fig. 6. The actor–critic neural network architecture.

policy function $\pi\ (a \parallel s; \theta)$ and updating the parameters θ using a gradient ascent on the expected return $E_\pi\ [\rho_t]$. Deep policy gradient (DPG) is one of the most common policy-based DRL algorithms, in which the parametrized model $\pi\ (a \parallel s; \theta)$ is based on a ANN that estimates the probability of taking an action a in a specific state s:

$$\pi\ (s, \theta) = (p\ (a|s, \theta))_{a \in A}\ . \tag{33}$$

Therefore, the ANN becomes a probability density function over its inputs, the states of the environment. The DPG is then reduced to solving the following optimization problem:

$$maximize\ E_\pi\ [\rho_t]\ .$$

In the following sections, we present the four DPG algorithms used in this work.

4.2.1. REINFORCE

In standard REINFORCE algorithms [57], the gradient ascent of $E\ [\rho_t]$ is achieved by updating the policy parameters θ in the direction of $\rho_t \nabla_\theta log\pi\ (a_t \parallel s_t; \theta)$, the unbiased gradient estimate of the objective function:

$$\nabla_\theta E_\pi\ [\rho_t] \propto \rho_t \nabla_\theta log\pi\ (a_t \parallel s_t; \theta)\ . \tag{34}$$

Therefore, an objective function can be defined as

$$L_{RE}\ (\theta) = \hat{E}_t\ [\rho_t log\pi\ (a_t \parallel s_t; \theta)]\ , \tag{35}$$

where $\hat{E}_t$ indicates the empirical average over a finite batch of samples in an algorithm that alternates between sampling and optimization. The learning process is summarized in Algorithm 2.

Algorithm 2: REINFORCE algorithm

1: Initialize network parameters θ randomly
2: Initialize training memory with capacity TM_{max}
3: Initialize agent memory
4: **For** episode = 1, Max_episodes **do:**
5: **For** t=0, D-1 **do**:
6: Get initial state s_t.
7: Take action a with ε-greedy policy-based on $\pi(s_t, \theta)$.
8: Receive new state s_{t+1} and reward R_t.
9: Store transition (s_t, a_t, R_t, s_{t+1}) in agent memory.
10: **End for**
11: Calculate ρ_t for each transition in the agent memory.
12 Store (s_t, a_t, ρ_t) in the training memory.
13: Reset agent memory
14: Reset environment
15: **If** training memory is full:
16: Perform a gradient ascent step on: $\boldsymbol{L_{RE}(\theta)}$ w.r.t to θ.
17: Reset training memory
18: **End for**

4.2.2. Actor–critic

The idea behind actor–critic methods is to use the value function information to assist in the policy update. Actor–critic methods [58] use two models, which may optionally share parameters:

- The critic model is used to calculate an estimate of the state value function $b_t\ (s_t; \theta) \approx V^\pi\ (s_t)$ that is used as baseline for updating the policy parameters. The critic model updates its parameters using basic supervised learning from past experiences.
- The actor model updates the policy parameters θ for $\pi\ (s, \theta)$ in the direction suggested by the advantage function A. The advantage function specifies the advantage of choosing an action a_t in a state s_t, over the expected value of s_t and is defined as

$$A\ (a_t, s_t) = Q\ (a_t, s_t) - V^\pi\ (s_t)\ . \tag{36}$$

The advantage can also be estimated using ρ_t and $b_t\ (s_t)$ as

$$\hat{A}\ (a_t, s_t) = \rho_t - b_t\ (s_t; \theta)\ , \tag{37}$$

because ρ_t is an estimate of $Q\ (a_t, s_t)$ and $b_t\ (s_t; \theta)$ is an estimate of $V^\pi\ (s_t)$. Therefore, the actor updates its parameters θ in the direction of $\hat{A}\ (a_t, s_t)\ \nabla_\theta log\pi\ (a_t \parallel s_t; \theta)$.

For the actor and critic models in this work, we used a shared ANN. The ANN, shown in Fig. 6, features two output layers that predict the policy π using SoftMax activation and the baseline

$b_t(s_t)$ using linear activation. The learning process uses a loss function that combines the critic and actor loss functions:

$$L^{AC}(\theta) = \hat{E}_t\left[\alpha_v L_v(\theta) + L_\pi(\theta)\right], \tag{38}$$

where α_v is the predefined learning rates for the value function's parameter updates, respectively. The definitions of L_π and L_v are

$$L_\pi(\theta) = -\hat{A}(a_t, s_t)\, log\pi(a_t \parallel s_t; \theta), \tag{39}$$

$$L_v(\theta) = \| V^\pi(s_t) - b_t(s_t; \theta) \|^2. \tag{40}$$

Interestingly, the value function in this case, $V^\pi(s_t)$, is calculated using the collected experience, by accumulating the discounted rewards received following a given policy π. In other words, $V^\pi(s_t)$ is simply the discounted return ρ_t. Therefore,

$$L_v(\theta) = \| \rho_t - b_t(s_t; \theta) \|^2. \tag{41}$$

The actor–critic algorithm is described in Algorithm 3.

Algorithm 3: Actor-Critic algorithm

```
Initialize network parameters θ randomly
Initialize training memory with capacity TM_max
Initialize agent memory
For episode = 1, Max_episodes do:
    For t=0, D-1 do:
        Get initial state s_t.
        Take action a with ε-greedy policy-based on π(s_t, θ).
        Receive new state s_{t+1} and reward R_t.
        Store transition (s_t, a_t, R_t, s_{t+1}) in agent memory.
    End for
    Calculate ρ_t for each transition in the agent memory.
    Store (s_t, a_t, ρ_t) in the training memory.
    Reset agent memory
    Reset environment
    If training memory is full:
        Calculate Â(a_t, s_t), log π(a_t||s_t; θ), L_π(θ) and L_v(θ) for the whole batch.
        Perform a gradient descent step on: L^AC(θ) w.r.t to θ.
        Reset training memory
End for
```

4.2.3. Proximal policy optimization (PPO)

PPO [59], is a variation of the standard actor–critic algorithm that addresses the problem of destructive, large policy updates. The aim is to constrain the policy updates to a small range to ensure that the updated policy does not move too far from the old policy, in which the training data were collected. Therefore, a clipped surrogate objective function is implemented, in which $log\pi(a_t \parallel s_t; \theta)$ is replaced by a probability ratio of the old policy to the new policy r_t, defined as

$$r_t(\theta) = \frac{\pi(a_t \parallel s_t; \theta)}{\pi(a_t \parallel s_t; \theta_{old})}. \tag{42}$$

This ratio is clipped inside a small interval around 1 to prevent large policy updates. The clipped surrogate objective function is defined as:

$$L^{CLIP}(\theta) = min\begin{pmatrix} r_t(\theta)\hat{A}(a_t, s_t), \\ clip\left(r_t(\theta), 1-\varepsilon_{clip}, 1+\varepsilon_{clip}\right)\hat{A}(a_t, s_t) \end{pmatrix}, \tag{43}$$

where *clip()* function is defined earlier in (10) and ε_{clip} is a hyperparameter that determines the policy update interval.

The final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective. In this scheme, the change in probability ratio is ignored only when it would cause the objective to improve and included when it would worsen the objective. Thus, in an actor–critic framework, the PPO loss function is defined as

$$L^{PPO}(\theta) = \hat{E}_t[\alpha_v L_v(\theta) - L^{CLIP}(\theta)] \tag{44}$$

The pseudocode of the PPO algorithm is presented in Algorithm 4.

Algorithm 4: PPO algorithm

```
Initialize network parameters θ randomly
Initialize training memory with capacity TM_max
Initialize agent memory
For episode = 1, Max_episodes do:
    For t=0, D-1 do:
        Get initial state s_t.
        Take action a with ε-greedy policy-based on π(s_t, θ).
        Receive new state s_{t+1} and reward R_t.
        Store transition (s_t, a_t, R_t, s_{t+1}) in agent memory.
    End for
    Calculate ρ_t for each transition in the agent memory.
    Store (s_t, a_t, ρ_t) in the training memory.
    Reset agent memory
    Reset environment
    If training memory is full:
        Store the current parameters as θ_old = θ_0 ← θ
        For k=0, Max_epochs do:
            Calculate Â(a_t, s_t), r_t(θ_k), L^CLIP(θ_k), and L_v(θ_k) for the entire batch.
            Perform a gradient descent step on L^PPO(θ) w.r.t to θ_k.
        End for
        Reset training memory
End for
```

4.2.4. Asynchronous Advantage Actor–Critic (A3C)

A3C [60] is a multi-thread version of the actor–critic method that aims at reducing the convergence time, widen the search area, and overcome the problem of highly correlated samples gathered by a single agent. In the A3C algorithm, multiple actor agents interact in parallel with copies of the environment and collect experience that is used by multiple learners to train the model. The A3C learning process is illustrated in Fig. 7. After each episode, the actor agents input the n-steps experience $(s_t, a_t, \rho_t, s_{D-1})$ into the training memory. When the maximum number of samples T_{max} is reached, the learner agent uses these samples to update the network parameters θ using the actor–critic loss function and resets the training memory.

4.2.5. Proposed variations

According to the experiments conducted in this work, the A3C algorithm suffered from experience losses due to the memory reset operation. Additionally, when the policy was updated to regions outside the regions from which the latest samples were

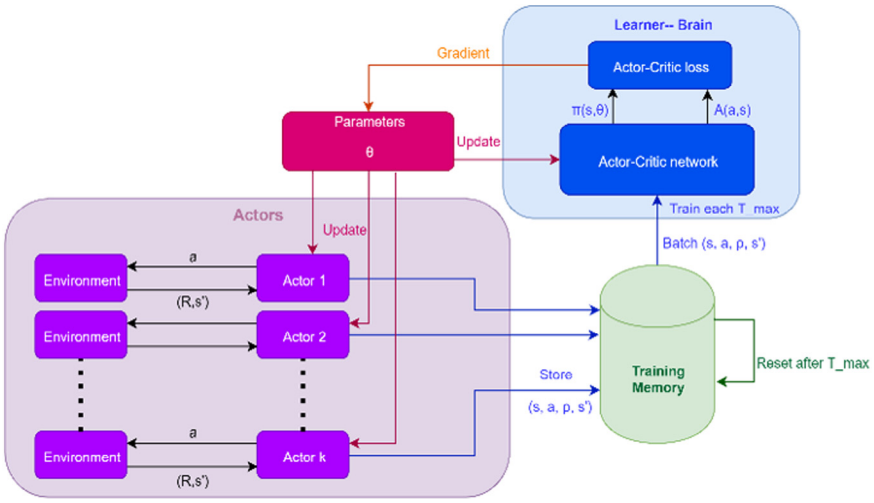

Fig. 7. Asynchronous Advantage Actor–Critic (A3C) framework.

collected, the accuracy of the model was damaged beyond recovery. In order to overcome these challenges, we introduced an experience replay technique similar to that introduced in the DQN algorithm. In the added experience replay, the learner agents update the parameters using random samples from all experiences collected during the entire process with a priority to new experiences. We also propose a semi-deterministic training in which the selection of an action a follows (45) instead of (33):

$$p\left(a = \underset{a'}{argmax}\left(\pi\left(a' \parallel s, \theta\right)\right)\right) = 1 - \varepsilon_k. \tag{45}$$

Allowing a deterministic selection of the action with the highest probability.

The following section details the improved performance and robustness of the algorithm achieved through this process. This variation is notably different from the deterministic DPG algorithm because it requires a stochastic training before the deterministic training, and it retains the ε-greedy strategy for exploration. We refer to this variation as A3C++ (A3C + experience reply + deterministic training and semi-deterministic training). The second variation is applied to PPO algorithm and was ran in a multi-thread framework, similar to the A3C algorithm and given the name PPO++ (PPO + semi-deterministic training).

5. Experimental implementation

5.1. Implementation details

We used an OpenAI toolkit called Gym [61] to implement the microgrid simulation. OpenAI Gym is an open-source toolkit for developing and comparing DRL algorithms. The parameters of the components used in the simulation are summarized in Table 1 and are based on the microgrid model and notations described in Section 2. The environment represents several days of energy management in which each episode corresponds to 1 day. For each episode, one of the 10 first days was chosen arbitrarily.

The DRL algorithms were implemented separately using Python, TensorFlow, and Keras [66]. DRL algorithms are sensitive to hyperparameters. Therefore, an empirical comparison between these parameters with changes in hyperparameters is implemented. We tested all the algorithms, except DQN, on various values of the speed of decay ε_{decay} as it is the common parameter between on-policy value-based algorithms and policy-based algorithms (Appendix Table A.1). The algorithms including a memory replay are tested on different values of the memory capacity

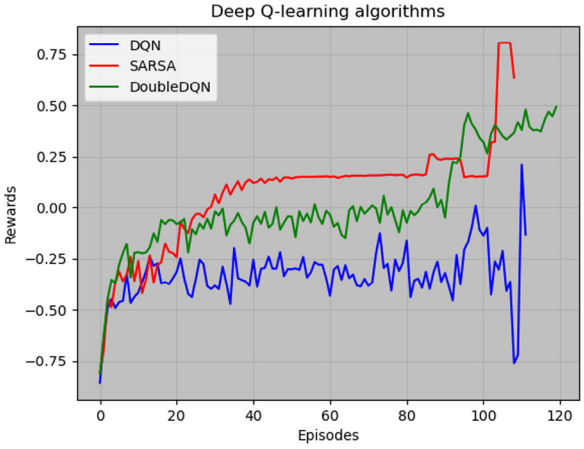

Fig. 8. Deep Q-learning algorithms: Learning curves. Average daily rewards over 10 days. Rewards are scaled by dividing the original reward function by 100.

TM_{max} and batch sizes (Appendix Table A.2). Finally, the actor–critic algorithms (actor–critic, A3C, PPO) are tested on different values of α_v (Appendix Table A.3). The algorithm-specific hyperparameters are chosen either according to the best results, e.g. TN for the doubleDQN algorithm (Appendix Table A.4), or according to their original papers e.g. $\varepsilon_{clip} = 0.2$ for PPO [59].

6. Results and discussion

6.1. Average rewards and training time

We ran the RL algorithms in the simulated environment and recorded both the training performances and the daily rewards averaged over 10 days. The learning processes are shown in Figs. 8 and 9 for each of the RL algorithms. The learning curves indicate high instability in the basic one-step DQN algorithm and a non-recoverable destruction of PPO algorithm. The remainder of the algorithms displays a satisfactory learning process that led to a reasonable convergence. The double DQN algorithm appears to have a stable learning curve, which supports the argument that the target network contributes advantageously to the stability of the DQN algorithm, as outlined in Section 4. Overall, deep Q-learning algorithms exhibit better learning stability than policy-based algorithms.

Table 1
Microgrid components and MDP parameters.

Parameter	Value
ESS parameters	
η_c	0.9
η_d	0.9
C_{max}	250 kW
D_{max}	250 kW
$B_{\max}$	500 kW
DER parameters	
G_t	1% of the hourly wind energy generation records from a wind farm in Finland (kW) [48].
C_{gen}	32 €/MW. The estimated Levelized Cost of Energy (LCoE) of the latest wind farm project in Finland [62].
Grid parameters	
P_t^d	Down-regulation prices in Finland (€/kW) [49]
P_t^u	Up-regulation prices in Finland (€/kW) [49]
$C_{\text{tr}_{\text{imp}}}$	9.7 €/MW Output from the grid and consumption fees [63]
$C_{\text{tr}_{\text{exp}}}$	0.9 €/MW Input into the grid and power plant fees [63].
TCL parameters	
N_{tcls}	100
T_t^0	Temperature records in Helsinki (°C) [64]
C_a^i	$\mathcal{N}(0.004, 0.0008)$
C_m^i	$\mathcal{N}(0.3, 0.004)$
q^i	$\mathcal{N}(0.0, 0.01)$
L_{TCL}^i	$\mathcal{N}(1.5, 0.01)$ (kW)
T_{min}^i	19.0 °C
T_{max}^i	25.0 °C
Residential load parameters	
N_L	150
$L_{b,t}$	Represented in Fig. 15.
λ_i	$\mathcal{N}(10, 6)$ (kW)
β_i	$\mathcal{N}(0.4, 0.3)$
General parameters	
D	24
δ_t	$\{-2, -1, 0, 1, 2\}$
cst	1.5
$threshold$	4
P_{market}	5.48 €/kW market price of February 2018 in Helsinki 2018 [65].
MDP parameters	
N_A	80
A_{tcl}	{0, 50, 100, 150}
A_P	$\{-2, -1, 0, 1, 2\}$
A_D	{ESS, Grid}
A_E	{ESS, Grid}
γ	1.0
t	1 h

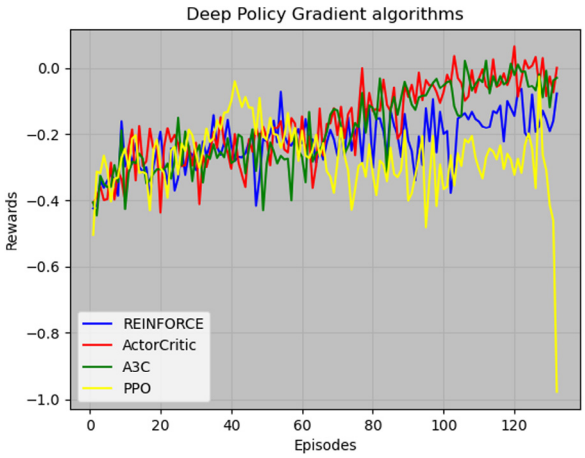

Fig. 9. Deep policy gradient algorithms: learning curves. Average daily rewards of 10 days.

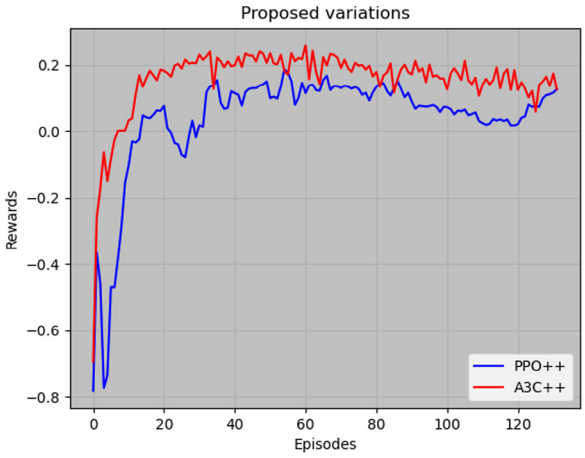

Fig. 10. A3C++ and PPO++. Average daily rewards of 10 days.

Table 2
Average total rewards after training.

Algorithm	Training time (min)	Reward 10^{-2}
1-step DQN	7.76	0.05
SARSA	7.78	0.16
Double DQN	11.30	0.13
REINFORCE	3.96	0.05
actor–critic	3.97	0.07
PPO (16 threads)	0.81	−0.08
PPO++	0.62	0.13
A3C (16 threads)	0.29	0.13
A3C++	0.75	**0.24**

Fig. 10 displays the learning processes of the proposed variations A3C++ and PPO++, in which the models were clearly able to find better optimal policies. A summary of the average total reward result from the test simulations and the training time for each algorithm is presented in Table 2.

6.2. Benchmarking the results

In order to evaluate the usefulness of the proposed microgrid's model and methods, we compare the results with two baselines:

A theoretical optimal controller: At each day, we run a sequential optimization algorithm over 24 h of the simulation, given all the information of production, consumption, prices, and temperatures. This is considered as a "theoretical optimal solution" because it is based on "perfect knowledge" of the variables and dynamics of the environment for the 24 h. The optimization is based on a genetic algorithm. This baseline will be used to evaluate the DRL algorithms' performances (see Fig. 12).

A theoretical retailer: A retailer buying the exact amount of electricity in the day-ahead market and selling it to the same customer base (loads and TCLs) in our simulation at the market price P_{market}. This is also a theoretical baseline as in practice, the quantities purchased in the day-ahead market are based on demand forecasts, and therefore, the retailer would usually need to balance their purchases in the intraday or the balancing market. This baseline is used to evaluate the proposed microgrid's model and its economic benefits. We use the day-ahead market prices [67] from the same period of the simulation.

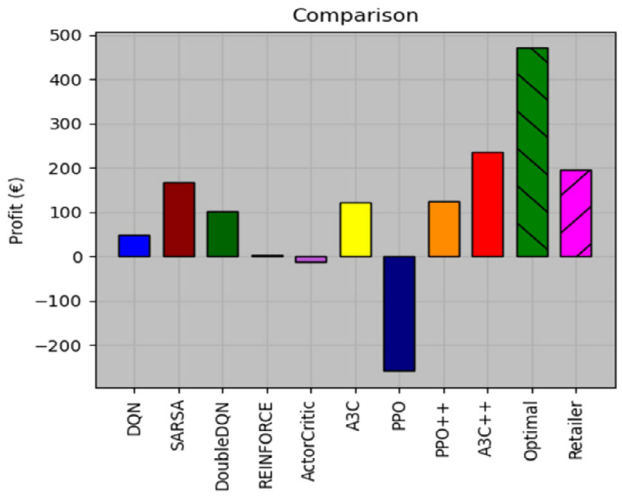

Fig. 11. Total profit of 10 days yielded by DRL algorithms, a theoretical optimal control of the microgrid, and a retailer supplying the same demand without microgrid.

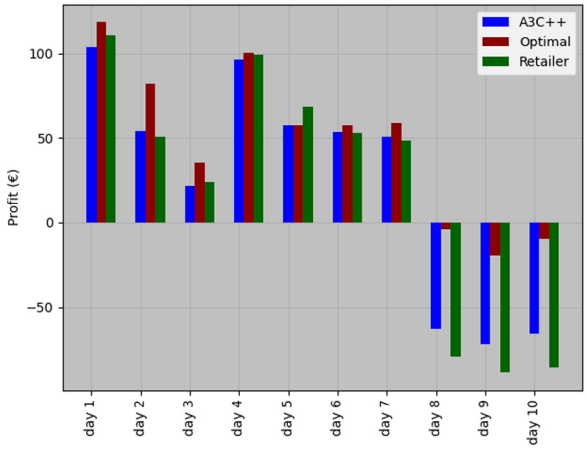

Fig. 12(a). Daily profit under A3C++, the optimal microgrid's control and an electricity retailer.

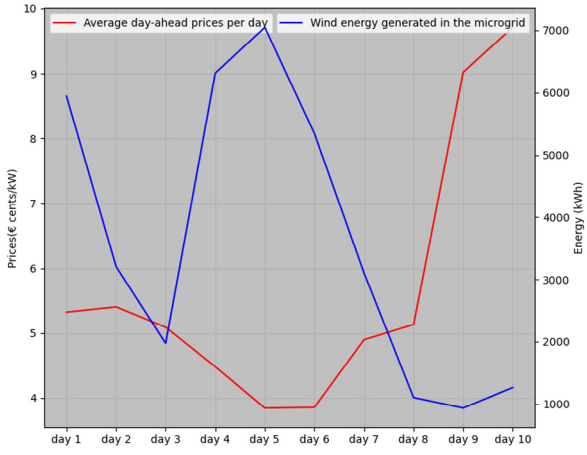

Fig. 12(b). Daily average electricity prices in the day-ahead market, and daily wind production in the microgrid.

The total 10 days' profits yielded by the DRL algorithms, the optimal controller, and the retailer are represented in Fig. 11. The prior observations indicate that the proposed variation A3C++ outperforms the rest of the algorithms presented in this paper, as well as the theoretical retailer. SARSA has in turn shown a good level of profit but not enough to excel the profit yielded by the retailer. A3C++ has yielded 50% of the optimal controller's profit and has outperformed the retailer by 24%. The success of A3C++ suggests that adding memory replay and a semi-deterministic training to the A3C algorithm can significantly improve the search quality, in turn leading to superior policy results.

The substantial difference between the retailer's profit and that of the optimal microgrid controller suggests a significant benefit of the proposed microgrid setup. This hypothesis can be validated by observing the results in different conditions of power generation and electricity prices. Fig. 12(a) shows the daily profits of A3C++, the optimal microgrid control, and the retailer for 10 consecutive days, in which, the power generated, and the day-ahead electricity prices differ significantly. The first observation is that the microgrid is more resilient to local energy scarcity than a retailer is to high electricity prices in the wholesale market. The optimal microgrid controller has outperformed the retailer in 9 out of 10 days. Therefore, the profitability of the proposed microgrid setup is higher than the proposed retailer. A3C++ algorithm has shown a very close performance to the optimal solution during the days with energy availability and outperformed the retailer in 6 out of 10 days. It is worth mentioning that customers pay a higher bill to the retailer, since the latter charges the TCLs the same price per kW as the other loads while the microgrid charges the TCLs a lower price in exchange of their direct control.

Fig. 12(b) shows an interesting negative correlation between the energy generated in the microgrid and the electricity prices in the day-ahead market. This observation suggests that the wind energy in the Nordic countries has reached a penetration level that allows it to influence the electricity market prices. The same observation was made in [68]. Even though the wind power data used in this study is specific to one production site in Finland, the correlation with the electricity prices is clear. This can be explained by the fact that most wind farms in Finland are located in the west coast and therefore, are subject to the same wind speed, which makes the wind power generation rather homogeneous. Therefore, the conclusions drawn about the economic advantage of the proposed microgrid and methods may need further verifications for different ecosystems.

6.3. Detailed single day behavior

In this section, we look closely at the behavior of the simulation environment under A3C++ algorithm compared to the optimal control solution. The behavior presented is according to the data of a single day presented in Fig. 13. However, the algorithms and simulations are available in [69], and the reader is invited to examine the results using different days of the simulation. The results related to the electricity sale prices, TCL energy allocation, and ESS state-of-charge are presented in Fig. 14. The energy purchased from the grid, and energy sold to the grid, are presented alongside the balancing market prices in Fig. 15.

The first notable result is that both A3C++ and the optimal controller have raised the sale prices at the early hours of the day even though the energy was available. This is an example a non-intuitive action, as the peak demand hour and the energy scarcity is latter in the day. However, this behavior becomes more reasonable when observing the amount of energy allocated to the TCLs, the energy stored in the ESS, and the energy sold to the grid at the beginning of the day. The demand from the price responsive loads had to be reduced in order to charge the battery, feed the TCLs and most importantly sell maximum amounts of energy to the grid at the highest price (∼10 cents/kW). The ESS charge and discharge curve of A3C++ is similar to that of the optimal

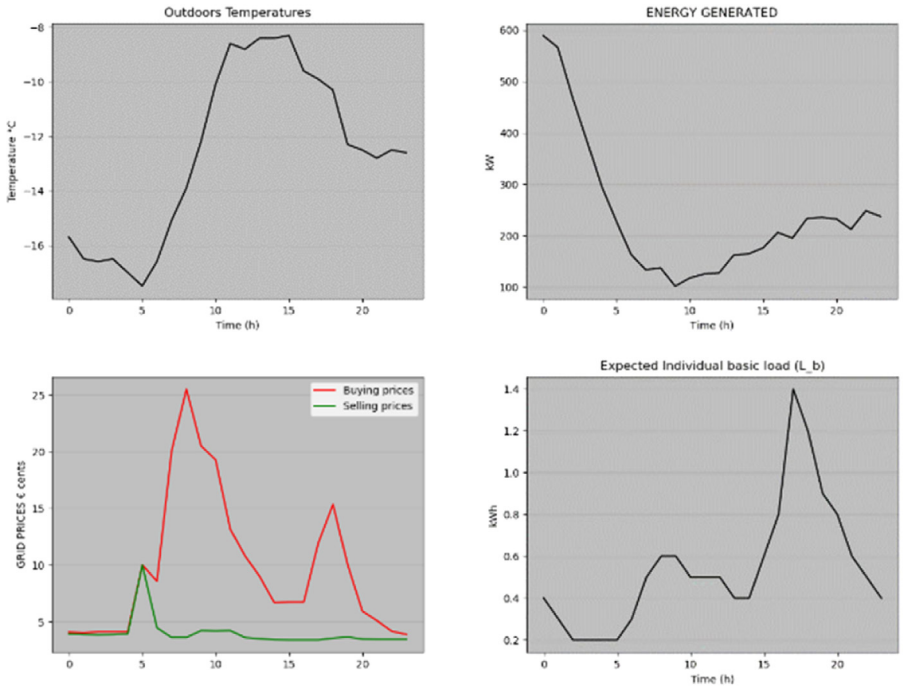

Fig. 13. Microgrid's environment data for one day of the simulation: outdoor temperatures, energy generated, grid prices, and expected individual basic load.

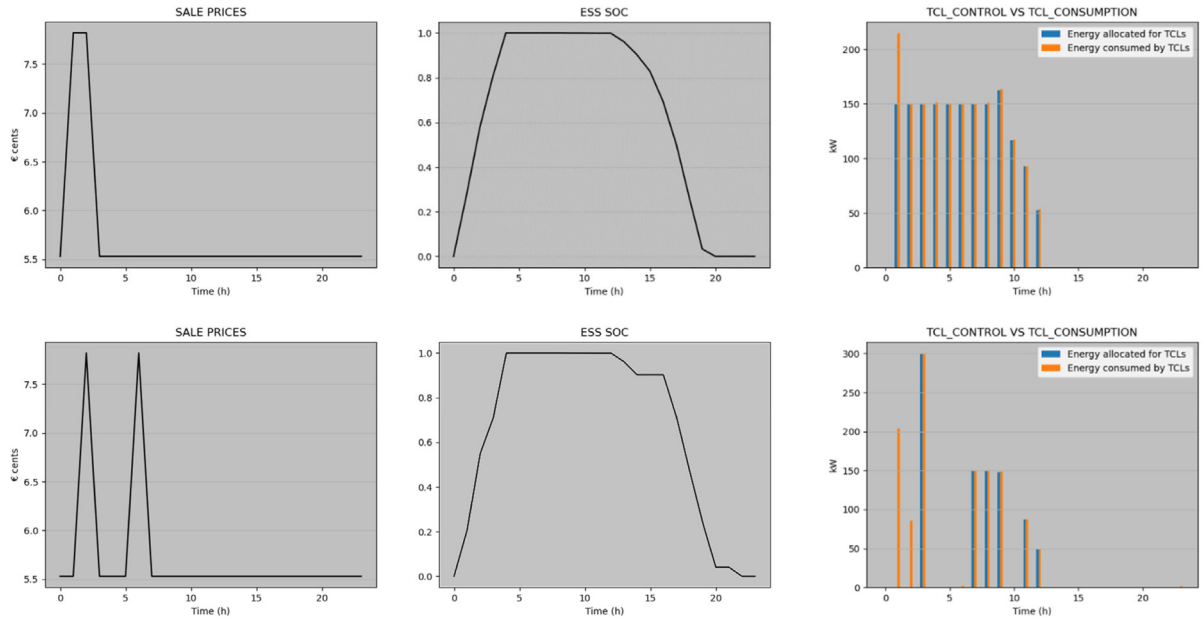

Fig. 14. Comparison between the A3C++ behavior (upper row) and the theoretical optimal control (lower row) during one day of the simulation.

controller. Basically, A3C++ has managed to anticipate the future and stored some energy in the ESS for later use. This behavior has allowed it to avoid buying high amounts of electricity at the time of high demand and low power generation. The energy allocated to the TCLs by A3C++ is a little different from the optimal policy. However, both have managed to feed the TCLs enough power, during the first half of the day, to maintain their temperature comfort for the rest of the day without requiring more energy. In fact, since the TCLs are equipped with backup controllers, they would consume more energy than what is allocated if that is necessary to maintain their indoor temperature comfort. This is shown in the case of the optimal control at the beginning of the day. This is a clear exploitation of the thermal flexibility of the TCLs, as they were successfully used as an energy storage system in the form of heat inside the houses.

The amounts of energy exchanged with the grid with regards to the up- and down-regulation prices represented in Fig. 15, show that both control methods have managed their sales and purchases in an optimal way. Both controllers have sold a maximum amount of energy at the time with highest selling price (5 am–6 am), and avoided buying big amounts of energy at the peak buying prices, despite the high demand and low generation at the evening (5 pm–7 pm). Even though A3C++'s sales to the grid are less than the optimal solution, the similarities between the two results are quite noticeable. Therefore, A3C++ was able to converge to near-optimal policy, even though it only observes one

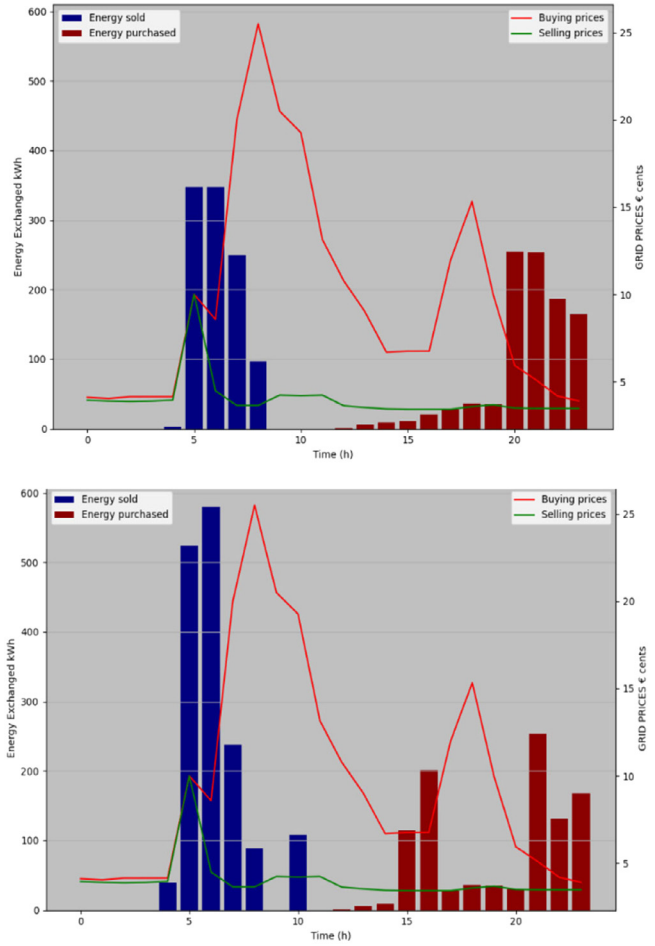

Fig. 15. Power exchanged with the grid and with regards to prices in the balancing market: a comparison between the results of A3C++(top) and those of the optimal policy (bottom).

hour's data at a time and has no prior knowledge of the system's dynamics and variables.

7. Conclusion

In this paper, we studied a multi-task EMS for a residential microgrid with multiple sources of flexibility. The microgrid model proposed considers the potential demand flexibility offered by price-responsive loads and TCLs. The proposed EMS coordinates between the ESS, the main grid, the TCLs, and the price-responsive loads to ensure optimal management of the local resources. The uncertain nature of the microgrid components and the high dimensionality of their variables incentivizes the use of intelligent learning-based methods in the EMS, such as DRL algorithms. In this paper, we also presented a comprehensive and experimental comparison of the current state-of-the-art DRL algorithms and proposed improved versions of the A3C and PPO algorithms that outperformed the existing algorithms. The numerical results show that DRL algorithms achieved different levels of convergence. The results revealed that adding an experience replay and a semi-deterministic training to the A3C algorithm improved the convergence and resulted in superior policies. The results were benchmarked against a theoretical optimal controller that has perfect knowledge of the system's variables and dynamics for the whole day, and an electricity retailer who buys electricity from the day-ahead market and supplies the same demand without a microgrid. The results show a significant advantage of the proposed microgrid's model and coordination algorithms compared to an electricity retailer, in terms of economic profitability and resiliency to hard conditions. The proposed A3C++ algorithm has achieved 50% of the theoretical optimum's profit and surpassed the retailer's performance by 24%.

Designing and implementing a successful EMS for future microgrids constitutes a challenging task because of the high dimensionality and uncertainty of the microgrid components. Though DRL methods have proved successful in the gaming field, they are far from perfect, and such methods face implementation difficulties in real problems due to their data inefficiency, instability, and slow convergence. Currently, concerted efforts are being made to improve the performance of DRL algorithms and enhance their applicability in real-world problems.

Further research based on the results of this paper can be extended in several directions. First, in the model of price-responsive loads, we used a simple model to describe the responsiveness of residential loads to the electricity prices. This model can be replaced by intelligent agents in future work, which would learn the optimal consumption considering the trade-off between comfort and the electricity bill, given the price uncertainty. The TCLs can also be modeled as intelligent price-responsive agents that adjust their power according to the price levels in future models. Secondly, in the control mechanisms, the ESS, and grid priorities were only determined in cases of energy deficiency or excess. However, charging and discharging of the ESS can also be achieved through exchanging electricity with the main grid or neighboring microgrids. Additionally, this model assumed that the electricity balancing from the grid was performed automatically in real-time using up- and down-regulation market prices. However, in reality, utility companies buy, and sell electricity in advance in the day-ahead and intraday markets using energy demand forecasts. Therefore, a DRL-based planning module can be added to anticipate the scarcity of energy and buy electricity in advance in the day-ahead or intraday markets. We plan to implement this model and investigate its potential in future work. Finally, we intend to study the ability of these algorithms to transfer their knowledge learned in simulations to a physical microgrid with a similar setup.

Table A.1
Comparison of DRL algorithms with variation of the speed of decay ε_{decay} related to the ε−**greedy** strategy.

ε_{decay}	SARSA	Double DQN	REINFORCE	actor–critic	PPO	PPO++	A3C++
1.0E−5	0.09	**0.13**	0.05	0.06	−0.92	**0.13**	0.21
5.0E−5	**0.16**	0.11	**0.05**	**0.07**	**−0.08**	0.03	**0.24**
1.0E−4	0.12	0.08	0.04	0.05	−0.15	0.04	0.19

Table A.2
Comparison of the memory replay-based DRL algorithms with variation of the size of memory. TM_{max}.

TM_{max}	Batch size	1-step DQN	SARSA	Double DQN	A3C++
300	100	0.05	0.01	0.12	0.17
	200	0.03	0.05	0.13	**0.24**
	300	−0.01	0.10	−0.01	0.11
500	100	0.00	0.07	**0.13**	0.19
	200	**0.05**	0.00	0.11	0.19
	300	−0.01	**0.16**	0.09	0.01
700	100	0.03	0.00	−0.01	0.23
	200	0.01	0.06	0.11	0.18
	300	0.05	0.07	0.00	0.14

Table A.3
Comparison of the actor–critic DRL algorithms with variation of α_v.

α_v	Actor–critic	A3C	A3C++	PPO	PPO++
0.09	0.04	0.07	0.02	−0.92	−0.08
0.2	−0.01	−0.03	0.16	**−0.08**	**0.13**
0.4	**0.07**	**0.14**	**0.24**	−0.92	−0.09
0.6	0.06	0.10	0.15	−0.92	−0.13
0.9	0.01	0.06	0.01	−0.95	−0.15

Table A.4
Target network update frequencies for double DQN algorithm.

TN	100	200	300	400	500
R	0.09	**0.13**	0.12	0.08	−0.28

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.

Acknowledgments

This work was supported by The Jenny and Antti Wihuri Foundation, Finland.

Appendix. DRL hyperparameters

See Tables A.1–A.4.

References

[1] Q. Jiang, M. Xue, G. Geng, Energy management of microgrid in grid-connected and stand-alone modes, IEEE Trans. Power Syst. 28 (3) (2013) 3380–3389, http://dx.doi.org/10.1109/tpwrs.2013.2244104.
[2] L. Guo, et al., Energy management system for stand-alone wind-powered-desalination microgrid, IEEE Trans. Smart Grid 7 (2) (2014) 1, http://dx.doi.org/10.1109/tsg.2014.2377374.
[3] 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, http://dx.doi.org/10.1109/tec.2014.2352554.
[4] X. Xu, H. Jia, H.-D. Chiang, D.C. Yu, D. Wang, Dynamic modeling and interaction of hybrid natural gas and electricity supply system in microgrid, IEEE Trans. Power Syst. 30 (3) (2015) 1212–1221, http://dx.doi.org/10.1109/tpwrs.2014.2343021.
[5] S. Krishnamurthy, T.M. Jahns, R.H. Lasseter, The operation of diesel gensets in a CERTS microgrid, in: 2008 IEEE Power Energy Soc. Gen. Meet. - Convers. Deliv. Electr. Energy 21st Century, 2008, pp. 1–8, http://dx.doi.org/10.1109/pes.2008.4596500.
[6] M. Tasdighi, H. Ghasemi, A. Rahimi-Kian, Residential microgrid scheduling based on smart meters data and temperature dependent thermal load modeling, IEEE Trans. Smart Grid 5 (1) (2014) 349–357, http://dx.doi.org/10.1109/tsg.2013.2261829.
[7] S.Y. Derakhshandeh, A.S. Masoum, S. Deilami, M.A.S. Masoum, M.E.H. Golshan, Coordination of generation scheduling with PEVs charging in industrial microgrids, IEEE Trans. Power Syst. 28 (3) (2013) 3451–3461, http://dx.doi.org/10.1109/tpwrs.2013.2257184.
[8] Y. Xu, W. Zhang, G. Hug, S. Kar, Z. Li, Cooperative control of distributed energy storage systems in a microgrid, IEEE Trans. Smart Grid 6 (1) (2015) 238–248, http://dx.doi.org/10.1109/tsg.2014.2354033.
[9] K.M.M. Huq, M.E. Baran, S. Lukic, O.E. Nare, An energy management system for a community energy storage system, in: 2012 IEEE Energy Convers. Congr. Expo., Vol. 1, 2012, pp. 2759–2763, http://dx.doi.org/10.1109/ecce.2012.6342532.
[10] C. Zhang, Y. Xu, Z.Y. Dong, J. Ma, Robust operation of microgrids via two-stage coordinated energy storage and direct load control, IEEE Trans. Power Syst. 32 (4) (2017) 2858–2868, http://dx.doi.org/10.1109/tpwrs.2016.2627583.
[11] T.A. Nakabi, P. Toivanen, Optimal price-based control of heterogeneous thermostatically controlled loads under uncertainty using LSTM networks and genetic algorithms, in: F1000Research, Vol. 8, 2019, p. 1619, http://dx.doi.org/10.12688/f1000research.20421.1.
[12] F. Ruelens, B.J. Claessens, P. Vrancx, F. Spiessens, G. Deconinck, Direct load control of thermostatically controlled loads based on sparse observations using deep reinforcement learning, 2017.
[13] C. Zhang, Y. Xu, Z.Y. Dong, K.P. Wong, Robust coordination of distributed generation and price-based demand response in microgrids, IEEE Trans. Smart Grid 9 (5) (2018) 4236–4247, http://dx.doi.org/10.1109/tsg.2017.2653198.
[14] L. Jian, H. Xue, G. Xu, X. Zhu, D. Zhao, Z.Y. Shao, Regulated charging of plug-in hybrid electric vehicles for minimizing load variance in household smart microgrid, IEEE Trans. Ind. Electron. 60 (8) (2013) 3218–3226, http://dx.doi.org/10.1109/tie.2012.2198037.
[15] H. Hao, B.M. Sanandaji, K. Poolla, T.L. Vincent, Aggregate flexibility of thermostatically controlled loads, IEEE Trans. Power Syst. 30 (1) (2015) 189–198, http://dx.doi.org/10.1109/tpwrs.2014.2328865.
[16] J.L. Mathieu, M. Kamgarpour, J. Lygeros, G. Andersson, D.S. Callaway, Arbitraging intraday wholesale energy market prices with aggregations of thermostatic loads, IEEE Trans. Power Syst. (2015) http://dx.doi.org/10.1109/TPWRS.2014.2335158.
[17] C. De Jonghe, B.F. Hobbs, R. Belmans, Value of price responsive load for wind integration in unit commitment, IEEE Trans. Power Syst. 29 (2) (2014) 675–685, http://dx.doi.org/10.1109/tpwrs.2013.2283516.
[18] Y. Zhang, L. Fu, W. Zhu, X. Bao, C. Liu, Robust model predictive control for optimal energy management of island microgrids with uncertainties, Energy 164 (2018) 1229–1241, http://dx.doi.org/10.1016/j.energy.2018.08.200.
[19] A. Parisio, E. Rikos, L. Glielmo, A model predictive control approach to microgrid operation optimization, IEEE Trans. Control Syst. Technol. 22 (5) (2014) 1813–1827, http://dx.doi.org/10.1109/tcst.2013.2295737.
[20] A. Parisio, E. Rikos, G. Tzamalis, L. Glielmo, Use of model predictive control for experimental microgrid optimization, Appl. Energy 115 (2014) 37–46, http://dx.doi.org/10.1016/j.apenergy.2013.10.027.
[21] S. Baldi, I. Michailidis, C. Ravanis, E.B. Kosmatopoulos, Model-based and model-free 'plug-and-play' building energy efficient control, Appl. Energy 154 (2015) 829–841, http://dx.doi.org/10.1016/j.apenergy.2015.05.081.
[22] M. Wiering, M. van Otterlo, Reinforcement Learning State-of-the-Art, Vol. 12, 2012.
[23] B. Mbuwir, F. Ruelens, F. Spiessens, G. Deconinck, Battery energy management in a microgrid using batch reinforcement learning, Energies 10 (11) (2017) 1846, http://dx.doi.org/10.3390/en10111846.
[24] P. Kofinas, G. Vouros, A.I. Dounis, Energy management in solar microgrid via reinforcement learning using fuzzy reward, Adv. Build. Energy Res. 12 (1) (2018) 97–115, http://dx.doi.org/10.1080/17512549.2017.1314832.
[25] B.C. Phan, Y.C. Lai, Control strategy of a hybrid renewable energy system based on reinforcement learning approach for an isolated microgrid, Appl. Sci. 9 (19) (2019) http://dx.doi.org/10.3390/app9194001.
[26] B.G. Kim, Y. Zhang, M. Van Der Schaar, J.W. Lee, Dynamic pricing and energy consumption scheduling with reinforcement learning, IEEE Trans. Smart Grid 7 (5) (2016) 2187–2198, http://dx.doi.org/10.1109/TSG.2015.2495145.
[27] S. Zhou, Z. Hu, W. Gu, M. Jiang, X.-P. Zhang, Artificial intelligence based smart energy community management: A reinforcement learning approach, CSEE J. Power Energy Syst. (2019) http://dx.doi.org/10.17775/CSEEJPES.2018.00840.
[28] P. Kofinas, A.I. Dounis, G.A. Vouros, Fuzzy Q-learning for multi-agent decentralized energy management in microgrids, Appl. Energy 219 (2018) 53–67, http://dx.doi.org/10.1016/j.apenergy.2018.03.017.
[29] E. Foruzan, L.K. Soh, S. Asgarpoor, Reinforcement learning approach for optimal distributed energy management in a microgrid, IEEE Trans. Power Syst. 33 (5) (2018) 5749–5758, http://dx.doi.org/10.1109/TPWRS.2018.2823641.
[30] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3) (1992) 279–292, http://dx.doi.org/10.1007/BF00992698.
[31] B.V. Mbuwir, D. Geysen, F. Spiessens, G. Deconinck, Reinforcement learning for control of flexibility providers in a residential microgrid, IET Smart Grid (2019) http://dx.doi.org/10.1049/iet-stg.2019.0196.
[32] T.P. Lillicrap, et al., Continuous control with deep reinforcement learning, in: 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., 2016.
[33] V. Mnih, et al., Human-level control through deep reinforcement learning, Nature (2015) http://dx.doi.org/10.1038/nature14236.
[34] D. Silver, et al., A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science (80-.) 362 (6419) (2018) 1140–1144, http://dx.doi.org/10.1126/science.aar6404.
[35] V. François-lavet, R. Fonteneau, D. Ernst, Deep reinforcement learning solutions for energy microgrids management, in: Eur. Work. Reinf. Learn., no. 2015, 2016, pp. 1–7.
[36] N. Ebell, F. Heinrich, J. Schlund, M. Pruckner, Reinforcement Learning Control Algorithm for a PV-Battery-System Providing Frequency Containment Reserve Power, 2018.
[37] V.-H. Bui, A. Hussain, H.-M. Kim, Double deep Q-learning-based distributed operation of battery energy storage system considering uncertainties, IEEE Trans. Smart Grid (2019) 1, http://dx.doi.org/10.1109/tsg.2019.2924025.

[38] B.J. Claessens, P. Vrancx, F. Ruelens, Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control, IEEE Trans. Smart Grid 9 (4) (2018) 3259–3269, http://dx.doi.org/10.1109/TSG.2016.2629450.
[39] T. Chen, W. Su, Local energy trading behavior modeling with deep reinforcement learning, IEEE Access 6 (2018) 62806–62814, http://dx.doi.org/10.1109/ACCESS.2018.2876652.
[40] A. Prasad, I. Dusparic, Multi-agent deep reinforcement learning for zero energy communities, in: ISGT Europe, 2019, http://dx.doi.org/10.1109/ISGTEurope.2019.8905628.
[41] H. Hua, Y. Qin, C. Hao, J. Cao, Optimal energy management strategies for energy internet via deep reinforcement learning approach, Appl. Energy (2019) http://dx.doi.org/10.1016/j.apenergy.2019.01.145.
[42] 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) http://dx.doi.org/10.3390/en12122291.
[43] N. Tomin, A. Zhukov, A. Domyshev, Deep reinforcement learning for energy microgrids management considering flexible energy sources, EPJ Web Conf. 217 (2019) 01016, http://dx.doi.org/10.1051/epjconf/201921701016.
[44] R. Lu, S.H. Hong, Incentive-based demand response for smart grid with reinforcement learning and deep neural network, Appl. Energy 236 (2019) 937–949, http://dx.doi.org/10.1016/j.apenergy.2018.12.061.
[45] T.A. Nakabi, K. Haataja, Pekka Toivanen, Computational intelligence for demand side management and demand response programs in smart grids, in: 8th Int. Conf. Bioinspired Optim. Methods their Appl. Paris, 2018, 2018.
[46] J.R. Vázquez-Canteli, Z. Nagy, Reinforcement learning for demand response: A review of algorithms and modeling techniques, Appl. Energy 235 (2019) 1072–1089, http://dx.doi.org/10.1016/j.apenergy.2018.11.002.
[47] E. Barbour, D. Parra, Z. Awwad, M.C. González, Community energy storage: A smart choice for the smart grid? Appl. Energy 212 (2018) 489–497, http://dx.doi.org/10.1016/j.apenergy.2017.12.056.
[48] Fortum Oy, Wind Farm data, Finland, 2018.
[49] Fingrid, Fingrid open datasets, 2019, [Online]. Available: https://data.fingrid.fi/open-data-forms/search/en/index.html. [Accessed: 12-Dec-2019].
[50] T.A. Nakabi, P. Toivanen, An ANN-based model for learning individual customer behavior in response to electricity prices, Sustain. Energy Grids Netw. 18 (2019) http://dx.doi.org/10.1016/j.segan.2019.100212.
[51] Residential electric rates & line items, 2019, [Online]. Available: https://austinenergy.com/ae/residential/rates/residential-electric-rates-and-line-items. [Accessed: 16-Dec-2019].
[52] M.L. Littman, Markov decision processes, in: International Encyclopedia of the Social & Behavioral Sciences, Elsevier, 2001, pp. 9240–9242.
[53] L.-J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Mach. Learn. 8 (3–4) (1992) 293–321, http://dx.doi.org/10.1007/bf00992699.
[54] D. Zhao, H. Wang, K. Shao, Y. Zhu, Deep reinforcement learning with experience replay based on SARSA, in: 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, 2017, http://dx.doi.org/10.1109/SSCI.2016.7849837.
[55] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in: 30th AAAI Conference on Artificial Intelligence, AAAI 2016, 2016, pp. 2094–2100.
[56] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, 2000, pp. 1057–1063.
[57] R.J. Willia, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3) (1992) 229–256, http://dx.doi.org/10.1023/A:1022672621406.
[58] V.R. Konda, J.N. Tsitsiklis, On actor-critic algorithms, SIAM J. Control Optim. 42 (4) (2003) 1143–1166, http://dx.doi.org/10.1137/S0363012901385691.
[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv e-prints, p. arXiv:1707.06347.
[60] V. Mnih, et al., Asynchronous methods for deep reinforcement learning, 2016, arXiv e-prints, p. arXiv:1602.01783.
[61] G. Brockman, et al., OpenAI gym, 2016, arXiv e-prints, p. arXiv:1606.01540.
[62] Fortum Oy, Levelized Cost of Energy, Finland, 2020.
[63] Fingrid, Grid Service Fees for 2020.
[64] Finnish meteorological institute, Weather observations, Kaisaniemi observation station Helsinki, 2019, [Online]. Available: https://en.ilmatieteenlaitos.fi/download-observations. [Accessed: 12-Dec-2019].
[65] Helsingin Energia Oy, Market price electricity price determination | Helen, 2020, [Online]. Available: https://www.helen.fi/en/electricity/electricity-products-and-prices/marketpriceelectricity/price-determination. [Accessed: 24-May-2020].
[66] F. Chollet, Keras documentation, in: Keras.Io, 2015, [Online]. Available: https://keras.io/.
[67] Nord Pool Spot, Market data | nord pool, 2018.
[68] T. Lusth, Wind power and the electricity wholesale price volatility an empirical study of the effect of wind power on the Swedish energy market, 2018.
[69] T.A. Nakabi, DRL for microgrid energy management, 2020, http://dx.doi.org/10.5281/zenodo.3598386.