Smart home's energy management applying the deep deterministic policy gradient and clustering

Ioannis Zenginis*, John Vardakas*, Kostas Ramantas*, and Christos Verikoukis†
* Iquadrat Informatica S.L., Barcelona, Spain
{izenginis, jvardakas, kramantas}@iquadrat.com
† CEID, University of Patras, Patras, Greece,
cveri@upatras.gr

***Abstract*—Smart buildings, equipped with controllable devices and energy management systems are expected to be substantial parts of the future energy grids. In this paper, a Reinforcement Learning (RL)-based method is developed for the energy scheduling of a smart home's energy storage system, which is also equipped with a photovoltaic system. The proposed scheme aims to minimize the electricity cost of the smart home; the overall problem is formulated as a Markov decision process, and it is solved by applying the Deep Deterministic Policy Gradient (DDPG). The main advantage of the proposed method is that increases the degree of similarity between the train set and the test set, through data clustering, achieving superior energy schedules than the existing RL-based approaches.**

***Index Terms*—energy management, reinforcement learning, deep deterministic policy gradient, smart home, storage system**

I. Introduction

The smart grid of the future will contain local energy networks, known as Microgrids (MGs), that comprise Distributed Energy Resources (DER), such as Renewable Energy Sources (RES), and controllable devices, such as Energy Storage Systems (ESSs) [1]. The ESS is a valuable component because it mitigates the impacts of fluctuations in the demand and the RES' generation, while it also provides with energy management flexibilities [2]. For example, under a dynamic pricing scheme, it can store energy during low price periods and provide it back over peak pricing [3].

The integration of MGs of various scales, from a single building to a large urban area [4], aims to make the operation of energy systems more efficient, economic and environment-friendly [5]. Therefore, energy management methods ought to be developed for the effective coordination of the various components included in MGs.

Traditionally, the MGs' energy management issue is formulated as an optimization problem that selects set points for the controllable devices so that various objectives are achieved. The Model Predictive Control (MPC) method is applied in [6] - [8] for the energy cost minimization, as well as in [9] for maximizing the electricity selling profit of a smart home. The optimal solutions in [6]-[9] are obtained through the utilization of commercial solvers. On the contrary, this is avoided in [10] - [12] by applying approximate dynamic programming, and modeling the MGs' operation as a Markov Decision Process (MDP).

The energy management solutions in [6]-[12] are model-based, which entails detailed knowledge of the MGs' dynamics to construct accurate models that describe their operation. In addition, precise predictions of stochastic variables are necessary to obtain the optimal control decisions. Any inaccuracies of the employed system models and the forecasting methods may deteriorate the solutions' quality.

RL methods, in contrast, can be used for developing energy management schemes that do not rely on detailed description of the MGs' dynamics, precise predictions of stochastic variables' values, and pricey commercial solvers. In RL, an agent learns, through interacting with an environment, what actions should take in given situations (states) in order to maximize numerical returns (rewards). Historical data are used for training the RL agents, which can then generalize the obtained knowledge and handle new situations in real-time.

Recently, numerous RL-based schemes have been proposed for energy management purposes. The Q-learning and the Deep Q-learning (DQN) algorithms are utilized in [13] - [15] and in [16]-[17], respectively, for minimizing the energy costs of smart buildings through their controllable appliances' smart scheduling. DQN is also employed in [18] for the minimization of the operation cost of a MG that comprises controllable DER, RES and ESSs, while a double DQN approach is adopted in [19] for optimizing the synergy of a MG with an external ESS.

When RL algorithms are applied, the problem of optimal energy scheduling, which is mainly described by continuous variables, is represented by an MDP. Nevertheless, Q-learning is appropriate for MDPs characterized by discrete state and action spaces, while DQN is appropriate for continuous state spaces and discrete action spaces. As a result, the aforementioned methods suffer from the curse of dimensionality [20]. To resolve the dimensionality issue, RL algorithms compatible with continuous state and action spaces have been developed in [21] - [27] for the real-time scheduling of controllable appliances in smart buildings, such as the trust religion policy optimization (TRPO) algorithm in [21] and the deep deterministic policy gradient (DDPG) algorithm in [22]-[27].

Contrary to [21]-[23], where the existence of energy sources is neglected, a DDPG-based energy management methodology is proposed in [24] for an islanded MG that contains a diesel generator, a PV system, and an ESS. DDPG-based control strategies are also applied in [25] and [26] with the objective to minimize the operating cost of residential buildings that contain both energy sources and storage units. However, the concept of real-time pricing is neglected in [24] - [26]. As opposed to that, dynamic pricing is considered in [27], where

a DDPG-based energy management scheme is employed to minimize the energy cost of a smart home.

Motivated by the interest of the scientific community on RL-based energy management, in this paper we propose a novel DDPG-based method for the real-time energy management of a smart home that contains a PV system and an ESS. The proposed methodology takes into account the uncertainties correlated with the load demand, the PV production and the electricity prices, and determines energy schedules for the ESS so that the smart home's electricity cost is minimized.

The main advantage of our proposed scheme is that it improves the effectiveness of the RL agents through data clustering. Concretely, instead of training a single agent based on a unified training set, as in ([13] - [27]), a clustering procedure is applied that divides the training set into data subsets (clusters), consisting of similar price curves. After that, a different DDPG agent is trained for each subset. The aforementioned data-clustering procedure is the first step towards increasing the degree of similarity between the train set and the test set. The second step includes a mechanism that matches every test day with the appropriate day-type subset, and hence, with the appropriate, already trained DDPG agent, which is then utilized for the accomplishment of the ESS's real-time energy scheduling.

The paper has the following structure: the various components of the smart home, and the RL formulation of the energy scheduling issue are described in Section II. The training process of the DDPG algorithm, the data-clustering procedure, as well as the accomplishment of real-time energy scheduling are described in Section III. Section IV includes the performance evaluation of our proposed approach, and Section V contains the conclusion.

II. System Model and RL Formulation

A smart home is considered that contains a PV system, an ESS, and an EMS, which is responsible for the power scheduling of the ESS so that electricity cost is minimized. The energy scheduling occurs over a decision horizon of T time slots t of duration Δt. Based on this convention, power and energy are used interchangeably in this study. At every time slot, the power balance is described as follows:

$$P_t^G = P_t^L - P_t^{PV} + P_t^{ESS} \tag{1}$$

where P_t^G denotes either the power imported, if $P_t^G \geq 0$, or the power exported to the main grid, if $P_t^G \leq 0$. P_t^L is the smart home's load, P_t^{PV} is the generated PV power, and P_t^{ESS} denotes either the power transferred to the ESS, if $P_t^{ESS} \geq 0$, or the power discharged from the ESS, if $P_t^{ESS} < 0$.

The ESS's State of Charge (SoC) SoC_t at t is given by:

$$SoC_t = SoC_{t-1} + \ m_t^{ESS} \left(P_t^{ESS} / N_{ESS} \right) \tag{2}$$

where N_{ESS} is the nominal capacity and m_t^{ESS} stands for the charging or discharging losses. During the ESS's charging, m_t^{ESS} is described by the charging efficiency n_c^{ESS} ($m_t^{ESS} = n_c^{ESS}$). In case the ESS is discharged, m_t^{ESS} is expressed as the reversed discharging efficiency n_d^{ESS}, i.e. $m_t^{ESS} = 1/n_d^{ESS}$. The state of charge SoC_t ranges within a minimum SoC_{min} and a maximum SoC_{max} level:

$$SoC_{min} \leq SoC_t \leq SoC_{max} \tag{3}$$

while P_t^{ESS} is bounded by a maximum power rate P_{max}^{ESS}:

$$\mid P_t^{ESS} \mid \leq P_{max}^{ESS} \tag{4}$$

In RL, an agent is trained in order to learn how to optimally interact with a specific environment. Concretely, after observing the current state of the environment, the agent takes a set of actions $a_t \in A_{s_t}$. The range of taken actions is constrained by the action space A_{s_t}, which depends on the environment's characteristics. Given the set of taken actions, a reward r_t is obtained, while the environment is led to the next state s_{t+1}. The sequence of states, actions and rewards during a decision horizon define an MDP episode, while the process of mapping an observed state to a set of taken actions consists the policy of the agent. The ultimate goal of the learning process is to find the optimal policy π^*. On that case, the performed actions at every given state, during a T-length MDP episode, maximize the total discounted reward R_t^π:

$$R_t^\pi = \sum_{t=0}^{T-1} r_t \gamma^t \tag{5}$$

where the discount factor γ ranges between 0 and 1, defining the significance of future rewards. Only the current reward signal is taken into account in case $\gamma = 0$, while in case $\gamma = 1$, the current and the future reward signals are equally important.

When the smart home's energy management is modeled as an MDP, the state of the system $s_t = \left\{ P_t^L - P_t^{PV}, SoC_t^{ESS}, \phi_t, t \right\}$ at time slot t is described by a set of variables that include the load P_t^L minus the PV generation P_t^{PV}, the ESS's state of charge SoC_t^{ESS}, as well as the electricity price ϕ_t and the time slot's incremental number t. In addition, the agent's action $a_t = \left\{ P_t^{ESS} \right\}, a_t \in A_{s_t}$ refer to the set points of the ESS's power P_t^{ESS}, while the action space A_{s_t} is defined by the operational constraints in (1)-(4). The reward r_t that the agent receives at every time interval is described by:

$$r_t = \begin{Bmatrix} -P_t^G \phi_t, & if \ P_t^G \geq 0 \\ -P_t^G \phi_t \rho, & if \ P_t^G < 0 \end{Bmatrix}, \tag{6}$$

which denotes that the reward is higher in case less amount of electricity is bought from the main grid ($P_t^G \geq 0$), or in case greater amount of electricity is sold ($P_t^G < 0$). It is also assumed that the price at which electricity is sold is a fraction of the buying price ϕ_t, i.e. $0 < \rho < 1$.

III. Proposed Energy Management Algorithm

The operation of the DDPG algorithm is based on the Q-function and the policy function. At any moment, a Q-function $Q(s, a)$ denotes the discounted reward when action a is performed in the current state s, given that the future actions are determined by a policy function $a = \pi(s)$ till the end of the decision horizon. DDPG uses four Deep Neural Networks (DNNs); a critic, which represent a Q-function with

parameters θ^Q, an actor, which represents the policy function with parameters θ^μ, as well as two target DNNs which are copies of the original networks, and their role is to make the training process more efficient [20]. The DNNs' parameters are updated until the optimal policy is achieved. The learning process occurs by taking into account MDP episodes i.e. days that have a T-length decision horizon. The stochastic variables' values are obtained from a hyperset of past data that contains load demand, PV production, and price datasets. The hyperset of historical data is first separated into subsets that consist of days with similar price profiles; after that, a different agent is trained for each one of the various subsets.

The learning procedure based on the data included in a day-type subset containing D days is presented in Algorithm 1. First, the DNNs' parameters and a replay buffer are initialized in lines 1 and 2, respectively. The algorithm runs for *M* episodes, where each episode is a randomly selected day. The main part of the agent's training process involves the following steps: first, the actor observes the current state s_t of the system and takes an action a_t to which noise Ξ_t is added for exploration (line 6). Given the performed action, the environment responds with a reward signal and moves to the next state (line 7). The transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer, from which a random mini-batch of *N* transitions $\left(s_\tau^{(i)}, a_\tau^{(i)}, r_\tau^{(i)}, s_{\tau+1}^{(i)}\right)$ is sampled (line 8). Next, a forward pass occurs at the critic for deriving the Q-values of the sampled state-action pairs $(s_\tau^{(i)}, a_\tau^{(i)}, \forall i \in N)$ (line 9), followed by a forward pass at the target Q-network for obtaining the Q-values of the pairs $(s_{\tau+1}^{(i)}, a_{\tau+1}^{'(i)}, \forall i \in N)$ (line 11), where $a_{\tau+1}^{'(i)}$ is determined by the target policy network (line 10). After that, the critic's parameters are updated in line 12 by minimizing the loss function $L\left(\theta^Q\right)$, while the actor's parameters are updated in line 13 by maximizing the expectation of the discounted reward $J\left(\theta^\mu\right)$. Eventually, the parameters of the target networks are updated in line 14 through slowly tracking the original networks' parameters.

Algorithm 2 describes the process of dividing the initial hyperset of data into subsets. It is based on the functionalities of the K-means algorithm, which partitions a dataset into K pre-defined clusters [28]. K-means assigns data-points to a cluster such that the squared Euclidean distance between the data-points and the cluster's centroid (arithmetic mean of all the data-points included in the cluster) is minimized. The optimal clusters' number is derived through utilizing the silhouette score, which is a measure of similarity of a data-point to the other data-points belonging to the same cluster, compared to the data-points in other clusters [29]. After finding k_ϕ price clusters (lines 2-6), the data hyperset is partitioned into subsets that contain similar price curves (lines 7-9). Afterwards, a different DDPG agent is trained for each subset by applying Algorithm 1.

The trained DDPG agents are utilized for the real-time energy scheduling of the smart home as described in Algorithm 3. An LSTM network, which is suitable for time-series forecasting [30], is first used for approximating the price curve of the next day (line 1). Following that, the approximated curve is matched with the closest price cluster (line 2), and then, the proper trained actor is applied (line 3) to perform real-time energy scheduling (lines 4-7).

Algorithm 1: DDPG agent's training process

1: Randomly initialize the critic's and actor's parameters θ^Q and θ^μ, respectively, and set the target networks' parameters equal to them: $\theta^{Q'} \leftarrow \theta^Q, \quad \theta^{\mu'} \leftarrow \theta^\mu$.
2: Define the size *B* of the replay buffer.
3: for $ep = 1$ to M :
4: Select a random day from D, observe s_0, initialize Ξ_t.
5: for $t = 0$ to $T-1$:
6: Observe s_t and perform action $a_t = \pi\left(s_t, \theta^\mu\right) + \Xi_t$.
7: Execute action a_t in the environment and observe the instant reward r_t and the next state s_{t+1}.
8: Store $(s_t, \quad a_t, \quad r_t, \quad s_{t+1})$ in the replay buffer, and sample from it a random mini-batch of *N* transitions $\left(s_\tau^{(i)}, \quad a_\tau^{(i)}, \quad r_\tau^{(i)}, \quad s_{\tau+1}^{(i)}\right)$, where $i \in N$ and $\tau \in (T-1)$.
9: Given $(s_\tau^{(i)}, a_\tau^{(i)}, \forall i \in N)$, do a forward pass of the critic for obtaining their Q-values $Q\left(s_\tau^{(i)}, a_\tau^{(i)}, \theta^Q\right), \forall i \in N$.
10: Given $s_{\tau+1}^{(i)}$, do a forward pass of the target policy network for obtaining the actions $a_{\tau+1}^{'(i)} = \pi'\left(s_{\tau+1}^{(i)}, \theta^\mu\right)$.
11: Given $(s_{\tau+1}^{(i)}, \ a_{\tau+1}^{'(i)})$, do a forward pass of the target Q-network for obtaining $Q'\left(s_{\tau+1}^{(i)}, \ a_{\tau+1}^{'(i)}, \quad \theta^{Q'}\right), \forall i \in N$.
12: Update the critic's parameters by minimizing $L\left(\theta^Q\right)$:

$$L\left(\theta^Q\right) = \frac{1}{N}\sum_i \left(y_\tau^{(i)} - Q\left(s_\tau^{(i)}, \ a_\tau^{(i)}, \ \theta^Q\right)\right)^2$$

$$y_\tau^{(i)} = r_\tau^{(i)} + \gamma \ Q'\left(s_{\tau+1}^{(i)}, \ a_{\tau+1}^{'(i)}, \quad \theta^{Q'}\right)$$

13: Update the actor's parameters by maximizing $J\left(\theta^\mu\right)$ using the following sampled policy gradient $\nabla_{\theta^\mu} J\left(\theta^\mu\right)$:

$$\nabla_{\theta^\mu} J(\theta^\mu) = \frac{1}{N}\sum_i \left(\nabla_{a_\tau^{(i)}} Q\left(s_\tau^{(i)}, a_\tau^{(i)}, \theta^Q\right) \nabla_{\theta^\mu} \pi\left(s_\tau^{(i)}, \theta^\mu\right)\right)$$

14: Update the target networks:

$$\theta^{Q'} \leftarrow \omega\, \theta^Q + (1-\omega)\, \theta^{Q'}, \theta^{\mu'} \leftarrow \omega\, \theta^\mu + (1-\omega)\, \theta^{\mu'}, \omega \ll 1$$

15: end
16: end

IV. Case Study

Our energy scheduling method is evaluated by considering a smart home that is equipped with a 3 kWp PV system and an ESS with nominal capacity N_{ESS}=6 kWh and maximum power rate P_{max}^{ESS}=3 kW. Table I reports the parameters' values considered for the rest of the smart home's components, as well as for the RL formulation of the problem. The decision horizon consists of T=24 time intervals of duration $\Delta t = 1$ hour, while stochastic variables' data for the load, the prices and the PV generation are obtained from [31], [32] and [33], respectively.

September and November are considered as test periods for evaluating our method. In both cases, data of one year

Algorithm 2: Derivation of data subsets

1: Input the hyperset that contains historical datasets Λ_Ω, Π_Ω and Φ_Ω consisting of Ω data-points, indexed by date, for the load, the PV production and the price, respectively.
2: for $k = 2$ to K :
3: Apply the K-Means on the price dataset to derive an array of labels $E_{\Phi k}$ that indexes to which of the k clusters each one of the Ω data-points is assigned, and an array $C_{\Phi k}$ that comprises the k clusters' centroids: $E_{\Phi k}, C_{\Phi k} = KMeans(\Phi_\Omega, k)$
4: Compute the silhouette score $S_{\Phi k}(\Phi_\Omega, E_{\Phi k})$
5: end
6: Keep the number of clusters k_ϕ for which the maximum silhouette score $S_{\Phi k_\phi}$ is obtained for the price dataset.
7: Apply the K-means on the price dataset for k=k_ϕ to obtain $E_{\Phi k_\phi}$ and $C_{\Phi k_\phi}$.
8: Given $E_{\Phi k_\phi}$, create k_ϕ subsets that are indexed by date and comprise price curves belonging to the same cluster.
9: Take the intersections of the indices of the obtained subsets with the indices of the load and PV datasets to derive the final subsets.

Algorithm 3: Real-time energy management

1: Estimate through an LSTM network the price curve of the next day using the previous G days' curves.
2: Compute the Euclidean distance between the estimated price curve and the clusters' centroids $C_{\Phi k_\phi}$ and match it with the closest price cluster $k_{\phi\_closest}$.
3: Load to the EMS the actor of the agent whose learning occurred based on the subset defined by $k_{\phi\_closest}$.
4: for $t = 0$ to $T - 1$:
5: Observe s_t and perform action a_t= $\pi(s_t, \theta^\mu)$.
6: Execute action a_t, and transit to the next state s_{t+1}.
7: end

before the test periods are utilized for determining the subsets required for training the DDPG agents. Through Algorithm 2, it is obtained that the 1-year-length datasets before both September and November are optimally partitioned into two price clusters (k_ϕ=2). Hence, two separate agents are trained that are correlated with the High Price (HP) and the Low Price (LP) subsets.

Every DDPG agent consists of two DNNs corresponding to the actor and the critic. Both of the DNNs have three hidden layers consisting of 400 and 300 neurons, respectively, and relu activation functions. The actor's input layer contains four neurons that represent the smart home's state s_t, while the output layer contains a single neuron that is activated by a tanh function, and then it is multiplied with the ESS's maximum power rate P_{max}^{ESS}. The critic's input layer contains five neurons since it takes as input both the system's state and the action, while the output layer contains a single neuron. The aforementioned architecture has been implemented in Pytorch [34]. The learning process for each agent lasts for about 2.5 hours being executed on a computer with an Intel Core i7 processor at 2.3 GHz and 8 GB RAM. The parameters' values taken into account for the learning procedure (Algorithm 1) are reported in Table I.

TABLE I: Parameters' values

ESS's parameters	n_c^{ESS}=n_d^{ESS}=0.95, SoC_{min}= SoC_0=0.2
Discount factor, electricity selling price factor, weighting factors	γ=1, ρ=0.5, ζ=0.5, β=0.4
Episodes' number, Replay buffer's size, Mini-batch size	M=15000, B=10^6, N=480
Noise parameters	ξ=0.15, m=0, dt=0.01, σ=0.2

For the implementation of the ESS's energy scheduling, the test days are assigned, before the beginning of the decision horizon, to the proper pre-trained agent. According to Algorithm 3, this is achieved through the assistance of an LSTM network, which estimates the price curve of the next day given the curves of the previous week i.e. $G = 7$. The input layer of the LSTM network contains $G \cdot T = 7 \cdot 24$ neurons that represent the past week's data. The input layer is connected with an LSTM layer consisting of 200 units, followed by a conventional layer containing 100 neurons. Finally, the LSTM network's output layer comprises $T = 24$ neurons that correspond to the estimated curve. One year of data are utilized for the training procedure, which takes place in Keras [35].

Table II reports the test days' assignment to the proper agent. The HP agent is utilized for the energy scheduling of 22 days in September, while the LP agent for the rest days. In November, only the HP agent is used because all days belong to the same price cluster. Our clustering-based method is compared with the case where clustering is neglected, and data of two months before each test period are utilized for the learning process of a single agent [27].

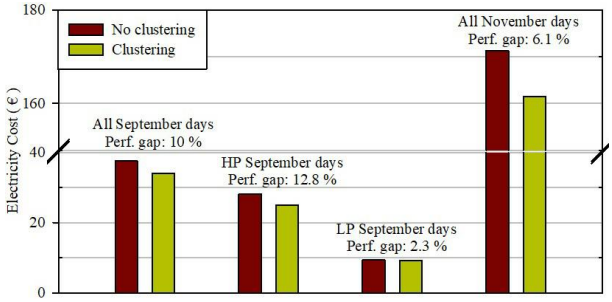

Fig. 1: Total electricity costs of the test days

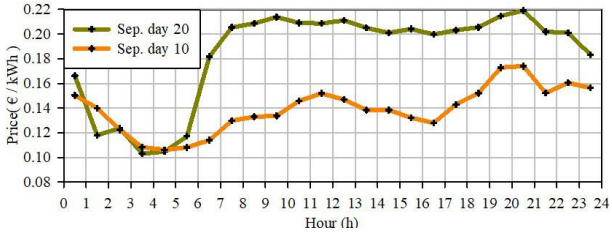

Fig. 2: Real-time prices

TABLE II: Assignment of test days to the price and temperature clusters

	High Price	Low Price
September	1, 2, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 19, 20, 21, 22, 23, 26, 27, 28, 29, 30	3, 4, 10, 11, 17, 18, 24, 25
November	All days	None

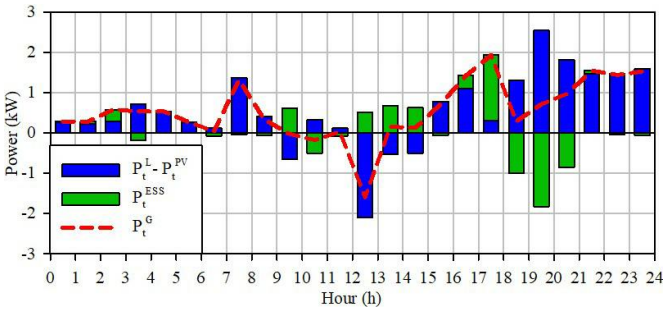

Fig. 3: Power scheduling for September's day 10 (LP agent)

Fig. 1 provides a comparison of the two methods (clustering vs no clustering) for the two test periods (September, November). In both cases, the proposed method performs better; the total electricity cost is by 10% and 6.1% lower for September and November, respectively. Our method's superiority is highlighted by the significantly lower cost (12.8%) that achieves in September's high-price days. This is due to the fact that all days in the training set, when our approach is implemented, belong to the high price cluster, while in contrast most of the days (49/62) comprising the training set of the no-clustering approach, belong to the low price level. With respect to November, where all days belong to the high price level, a lower performance gap (6.1%) is observed because the training set of the no-clustering method mainly consist of high-price days (45/61).

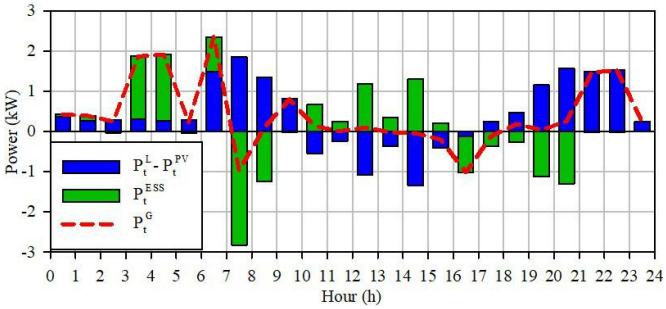

Fig. 4: Power scheduling for September's day 20 (HP agent)

Next, the power scheduling during two September days, characterized by different price levels, is compared. Fig. 2 reports the electricity prices for the test days. The ESS's power P_t^{ESS}, the load minus the PV power $P_t^L - P_t^{PV}$, as well as the power P_t^G exchanged with the main grid on the 10^{th} of September, when the energy management is carried out by the LP agent is presented in Fig. 3. The ESS is mainly recharged during 12:00-18:00 taking advantage of the high PV generation ($P_t^L - P_t^{PV} \leq 0$ during 12:00-15:00) and the relatively low electricity prices over this period. The stored energy is provided back during intervals characterized by higher prices, and by relatively high demand (18:00-21:00).

A quite different operation plan for the ESS is obtained for the 20^{th} of September (Fig. 4) by the HP agent. On that occasion, a significant amount of energy is transferred to the ESS during the low price period 03:00-07:00, and it is offered back to the demand during 07:00-09:00, period that is characterized by both higher load demand and prices. After that, the ESS is recharged again during 10:00-16:00, when there is available amount of energy excess (the demand is lower than the PV energy). Most of the stored energy is later provided back to the demand during 19:00-21:00, which is the peak-price interval of that day.

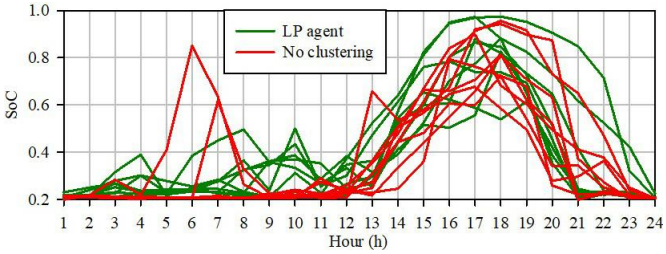

Fig. 5: ESS's schedules for low-price September days

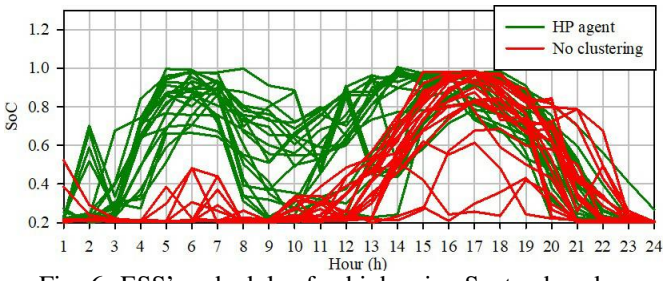

Fig. 6: ESS's schedules for high-price September days

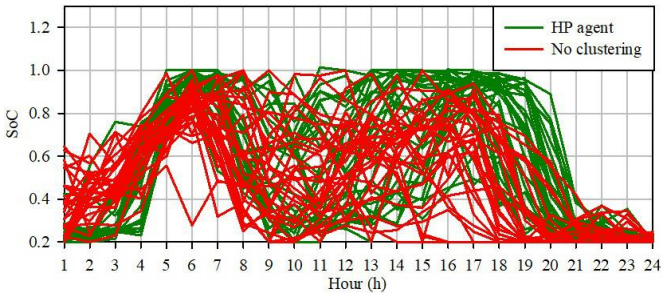

Fig. 7: ESS's schedules for all November days

The outcomes of Figs. 3 and 4 are special cases of general scheduling patterns. According to Fig. 5, the LP agent, which is responsible for the energy management of 8 of the September days, tends to recharge the ESS in the afternoon and discharge it in the evening. On the other hand, there is an additional recharging of the ESS early in the morning, followed by a morning discharging for the 22 September days whose scheduling is carried out by the HP agent (Fig. 6). Contrary to the different scheduling patterns obtained by the LP and the HP agents, the ESS's operation plans of low-price

and high-price days do not show a significant difference under the no-clustering approach.

The ability to determine different operation plans for the ESS, depending on the price profiles of the test days, is the main factor that leads to lower electricity costs in September, under our clustering-based approach compared to the no-clustering one. The performance gap between the two approaches is higher for test days that are characterized by high prices. As already noted, this is owing to the fact that the majority of days (49/62) in the training set of the no-clustering method are characterized by low prices. In November, the majority of days (45/61) in the training set of the no-clustering approach belong to the same price level with the test days. As a result, the two methods achieve similar ESS's scheduling patterns (Fig. 7). Yet, the proposed clustering-based approach outperforms the no-clustering candidate achieving by 6.1% lower electricity costs.

V. Conclusion

An RL scheme has been developed for the energy management of a smart home. The objective of the proposed scheme is to minimize the electricity cost by selecting appropriate set points for the smart home's storage system in real-time. The problem is formulated as an MDP, and it is solved by applying the DDPG algorithm, which is suitable for continuous state and action spaces. The main advantage of our clustering-based method compared to the existing RL-based methods is that increases the similarity degree among the price curves included in the training sets and those included in the test sets. This in turn leads to the derivation of more efficient operation plans for the storage device. In the future, we are planning to apply our method on systems that contain multiple energy sources, as well as multiple controllable devices.

References

[1] I. Zenginis, et al., "Optimal power equipment sizing and management for cooperative buildings in microgrids," *IEEE Trans. Ind. Inform.*, vol.15, no. 1, pp. 158-172, May 2018.

[2] X. Luo, et al., "Overview of current development in electrical energy storage technologies and the application potential in power system operation," *Appl. Energy*, Vol. 137, pp. 511-536, Jan. 2015.

[3] K. Bradbury, L. Pratson, and D. Patiño-Echeverri, "Economic viability of energy storage systems based on price arbitrage potential in real-time US electricity markets," *Appl. Energy*, vol. 114, pp. 512-519, Feb. 2014.

[4] J. S. Vardakas, et al., "Electricity savings through efficient cooperation of urban buildings: the smart community case of Superblocks in Barcelona," *IEEE Com. Mag.*, Vol. 56, no. 11, pp. 102-109, Nov. 2018.

[5] I. Zenginis, et al., "Cooperation in microgrids through power exchange: An optimal sizing and operation approach," *Appl. Energy*, Vol. 203, pp. 972-981, Oct. 2017.

[6] A. Parisio, E. Rikos, G. Tzamalis, and L. Glielmo, "Use of model predictive control for experimental microgrid optimization," *App. Energy*, Vol. 115, pp. 37-46, Feb. 2014.

[7] A. Parisio, E. Rikos, and L. Glielmo, "A model predictive control approach to microgrid operation optimization," *IEEE Trans. Contr. Syst. Techn.*, vol. 22, no. 5, pp. 1813-1827, Jan. 2014.

[8] Y. Zhang, et al., "Robust model predictive control for optimal energy management of island microgrids with uncertainties," *Energy*, vol. 164, pp. 1229-1241, Dec. 2018.

[9] L. Langer, and T. Volling "An optimal home energy management system for modulating heat pumps and photovoltaic systems," *Appl. Energy*, vol. 278, pp. 1156-1161, Nov. 2020.

[10] C. Keerthisinghe, G. Verbič, and A. C. Chapman, "A fast technique for smart home management: ADP with temporal difference learning," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3291-3203, Nov. 2016.

[11] H. Shuai, et al., "Stochastic optimization of economic dispatch for microgrid based on approximate dynamic programming," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 2440-2452, Jan. 2018.

[12] H. Shuai, J. Fang, X. Ai, J. Wen, and H. He, "Optimal real-time operation strategy for microgrid: An ADP-based stochastic nonlinear optimization approach," *IEEE Trans. Sust. Energy*, vol. 10, no. 2, pp. 931-942, Jul. 2018.

[13] S. Kim, and H. Lim, "Reinforcement learning based energy management algorithm for smart energy buildings", *Energies*, vol. 11, no. 8, Aug. 2018.

[14] R. Lu, S. H. Hong, and M. Yu, "Demand response for home energy management using reinforcement learning and artificial neural network," *IEEE Trans. Smart Grid*, vol 10, no. 6, pp. 6629-6639, Apr. 2019.

[15] X. Xu, et al., "A multi-agent reinforcement learning-based data-driven method for home energy management," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 3201-3211, Feb. 2020.

[16] E. Mocanu, et al., "On-Line Building Energy Optimization Using Deep Reinforcement Learning," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3698-3708, May 2019.

[17] P. Lissa, et al., "Deep reinforcement learning for home energy management system control," *Energy and AI*, vol. 3, Mar. 2021.

[18] Y. Ji, et al., "Real-time energy management of a microgrid using deep reinforcement learning, " *Energies*, vol. 12, no. 12, Jan. 2019.

[19] V. H. Bui, A. Hussain, and H. M. Kim, "Double deep Q-learning-based distributed operation of battery energy storage system considering uncertainties," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 457-469, Jun. 2019.

[20] T. P. Lillicrap, et al., "Continuous control with deep reinforcement learning," 2015.

[21] H. Li, Z. Wan, and H. He, "Real-time residential demand response, " *IEEE Trans. Smart Grid*, vol. 11, no. 5, pp. 4144-4154, Mar. 2020.

[22] Y. Du, et al., "Intelligent multi-zone residential HVAC control strategy based on deep reinforcement learning," *Appl. Energy*, vol. 281, Jan. 2021.

[23] G. Gao, J. Li, and Y. Wen, "DeepComfort: Energy-Efficient Thermal Comfort Control in Buildings via Reinforcement Learning, " *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8472-8484, May 2020.

[24] L. Lei, et al., "Dynamic Energy Dispatch Based on Deep Reinforcement Learning in IoT-Driven Smart Isolated Microgrids," *IEEE Internet Things J.*, vol. 8, no. 10, Dec. 2020.

[25] Y. Ye, et al., "Model-free real-time autonomous control for a residential multi-energy system using deep reinforcement learning, " *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 3068-3082, Feb. 2020.

[26] I. Zenginis, J. Vardakas, K. Ramantas, and C. Verikoukis, "Real-time energy management of a smart home based on deep deterministic policy gradient " in *Proc. EEEIC*, 2021.

[27] L. Yu, et al., "Deep reinforcement learning for smart home energy management, " *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2751-2762, Dec. 2019.

[28] X. Jin and J. Xan "K-Means Clustering, " *Springer*, pp. 563-564, 2010.

[29] K. R. Shahapure and C. Nicholas "Cluster quality analysis using silhouette score, " *Proc. IEEE DSAA*, pp. 747-748, 2020.

[30] J. Brownlee, "Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python," *Machine Learning Mastery*, 2018.

[31] OPSD. [Online]. Available: https://data.open-power-system-data.org/household_data/, acc. May 25, 2022

[32] Zenodo. [Online]. Available: https://sandbox.zenodo.org/record/632147#.YMuWub4zaUm, acc. May 25, 2022

[33] SOLARGIS. [Online]. Available: https://solargis.com/products/evaluate/useful-resources, acc. May 25, 2022

[34] PyTorch. [Online]. Available: https://pytorch.org/, acc. May 25, 2022

[35] Keras. [Online]. Available: https://keras.io/, acc. May 25, 2022