Residential Energy Management with Deep Reinforcement Learning

Zhiqiang Wan*, Hepeng Li†, and Haibo He*
*Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island
RI 02881 USA, Email: {zwan, he}@ele.uri.edu
†Lab. of Networked Control Systems, Shenyang Institute of Automation, Chinese Academy of Sciences
Shenyang 110016 China, Email: cn.h.li@ieee.org

***Abstract*—A smart home with battery energy storage can take part in the demand response program. With proper energy management, consumers can purchase more energy at off-peak hours than at on-peak hours, which can reduce the electricity costs and help to balance the electricity demand and supply. However, it is hard to determine an optimal energy management strategy because of the uncertainty of the electricity consumption and the real-time electricity price. In this paper, a deep reinforcement learning based approach has been proposed to solve this residential energy management problem. The proposed approach does not require any knowledge about the uncertainty and can directly learn the optimal energy management strategy based on reinforcement learning. Simulation results demonstrate the effectiveness of the proposed approach.**

I. Introduction

According to the latest International Energy Outlook 2017 [1], the world energy consumption is projected to increase 28% by 2040. Apart from installing new power plants, demand response (DR) provides an economically efficient way to alleviate the increasingly tense electricity demand. By shifting electricity consumption from on-peak hours to off-peak hours, DR offers a chance for the end-consumers to participate in operation and energy management of electric power grids to improve energy efficiency. In general, real-time electricity price is used by the electric system planner and operator to incite the consumers to participate in the DR programs.

For residential DR programs, energy management system plays an essential part in automatical response to real-time electricity prices and management of household appliances consumptions. A lot of studies in the literature have been focused on developing efficient DR policies for optimal residential energy manage (REM). To name a few, in [2], a learning-based DR strategy for optimal control of a heating ventilation and air conditioning system (HVAC) is developed to minimize the electricity costs. In [3], a DR strategy for scheduling of aggregated residential HVACs is investigated. In [4], the DR capability of electric water heaters (EWH) is evaluated for load-shifting and balancing reserve. In [5], [6], co-optimization of several types of home appliances, including controllable and shiftable appliances, is studied. In [7], a mixed integer linear or nonlinear programming model is proposed to make optimal DR schedules of different types of appliances while considering the users' comfort. In [8], deep Q-learning and deep policy gradient are applied to control the flexible loads such that the electricity cost and the load peak can be reduced. However, these works did not consider the value of battery energy storage in REM for DR programs.

Many researchers have investigated the REM problem while considering battery energy storage. Abdulgader et al. [9] analyze a smart home scenario with battery energy storage. An evolution algorithm Grey Wolf Optimizer based on the swarm intelligence is applied to manage the energy storage system by charging the battery when the electricity price is cheap and discharging it when the price is high. Melhem et al. [10] investigate the energy consumption and production of a smart home where the integration of the battery energy storage, the renewable energy sources, and the electric vehicle is considered. The REM is optimized by a mixed integer linear programming such that the electricity cost is minimized. Wei et al. [11] report an error-tolerant iterative adaptive dynamic programming algorithm to optimize the battery energy storage management in a smart home with Photovoltaics (PV) panels.

Recently, deep neural network (DNN) has been used in numerous applications [12]–[17], including image recognition, visual question answering, and machine translation. Deep reinforcement learning (RL), combinatin of DNN and RL, has obtained significant success in complex decision-making processes [18]–[23]. For example, a deep Q-network is developed in [18] to learn to make successful decisions only based on image observations. The discriminative features of the image observation are extracted by a convolutional neural network. The proposed deep Q-network is evaluated on the Atari games and its performance is comparable to that of a professional player. The great success of AlphaGo [19], defeating 18-time world champion Lee Sedol by 4 games to 1, is contributed by the integration of deep RL and Monte Carlo tree search. Despite the above perfect information applications, deep RL obtained promising results in imperfect information applications. Heinrich et al. [20] proposed Neural Fictitious Self-Play (NFSP) which combined deep RL with fictitious self-play to learn approximate Nash equilibrium for imperfect information games. NFSP has been successfully applied in two-player zero-sum games, such as Leduc poker and Limit Texas Holdem. DeepStack introduced in [21] defeated professional poker players in Heads-up No-Limit Holdem Poker by integrating recursive reasoning to tackle information asymmetry.

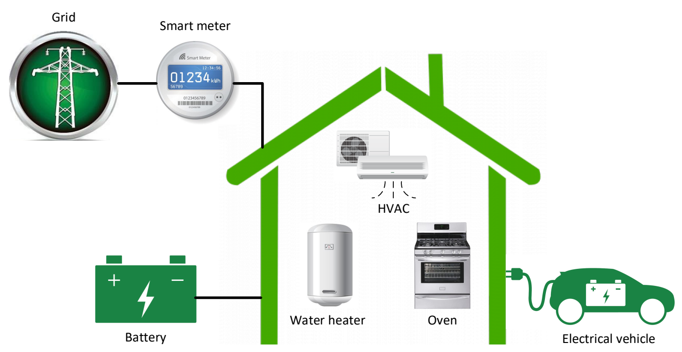


Fig. 1. The schematic diagram of the REM system.

In this paper, a deep RL based approach is proposed to solve the REM problem. This problem is formulated as a finite Markov decision process (MDP) process. The objective of this REM problem is to determine an optimal sequence of actions such that the total electricity cost is minimized. It is worth noting that the proposed approach is model-free which does not require any knowledge about the uncertainties of the home load and real-time electricity price. The proposed model consists of two networks, an actor network and a critic network. The actor network generates an action which determines the amount of electricity to be purchased from the utility. The critic network assesses the performance of executing this action. In these two networks, gated recurrent unit (GRU) network is applied to extract discriminative features from the input time-series data. Simulation results under real-world scenario verify the effectiveness of the proposed approach.

The remainder of this paper is organized as follows. Section II provides the problem formulation. Then, Section III introduces the proposed approach. Its effectiveness is verified by simulation results in Section IV. Lastly, a conclusion is given in Section V.

II. Problem Formulation

A. Environment Setup

A smart REM system is introduced in [24]. The schematic diagram of the REM system is presented in Fig. 1. The REM system consists of the power grid, the energy storage (battery) system, and the home loads. At each hour, the REM system decides the amount of electricity to be purchased from the utility. We consider a real-world scenario where the electricity price and home loads vary hourly. The real-world hourly electricity price is from the California ISO [25]. The household consumption is generated based on user behavior modeling in accordance with [7].

B. Problem Formulation

A finite MDP with discrete time step $t \in (1, 2, ..., 24)$ is applied to model the sequential decision-making process of this REM problem in the horizon of one day. The MDP is defined as the five-tuple $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}_{\cdot}(\cdot,\cdot), \boldsymbol{R}_{\cdot}(\cdot,\cdot), \gamma)$, where $\boldsymbol{S}$ denotes the system state, $\boldsymbol{A}$ is the set of feasible actions, $\boldsymbol{P}_{\cdot}(\cdot,\cdot)$ represents the system state transition probability, $\boldsymbol{R}_{\cdot}(\cdot,\cdot)$ is the immediate reward, and γ is a discounted factor.

1) State: The system state s_t consists of three kinds of information: (1) past N-hour electricity prices $(P_{t-N}, \dots, P_t)$; (2) past N-hour home loads $(L_{t-N}, \dots, L_t)$; (3) remaining battery energy E_t.

2) Action: The action a_t is the amount of electricity to be purchased from the utility at hour t. The purchased electricity can be used to power the home loads and charge the battery.

3) State Transition: The state transition from the state s_t to s_{t+1} is denoted as

$$s_{t+1} = f(s_t, a_t, \omega_t), \tag{1}$$

where the state transition is determined by the action a_t and the randomness ω_t. Specifically, the remaining battery energy is determined by the battery model $E_{t+1} = E_t + a_t - L_t$. However, the transition of electricity price P_t and home loads L_t are subject to randomness because of the lack of information about future prices and future loads.

4) Reward: The reward is defined as

$$r_t = \begin{cases} -P_t * a_t, & t \neq t_{24} \\ -P_t * a_t - \lambda * [\max(E_{min} - E_t, 0)], & t = t_{24} \end{cases} \tag{2}$$

where $P_t * a_t$ represents the electricity cost at hour t, E_{min} denotes the allowed minimum battery energy, and λ is a penalty factor.

In this problem formulation, an action-value function $Q_{\boldsymbol{\pi}}(s, a)$ is used to assess the performance of executing the action a under the state s. $Q_{\boldsymbol{\pi}}(s, a)$ is defined as the expected sum of discounted future rewards starting from state s, executing the action a, and thereafter following the policy π, i.e.,

$$Q_{\boldsymbol{\pi}}(s, a) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{k=0}^{K} \gamma^k * r_{t+k} \middle| s_t = s, a_t = a \right], \tag{3}$$

where π is the policy which maps from state s to the probability $\pi(a \mid s)$ of taking action a when given state s. In Eq. (3), K denotes the number of future time steps, and γ is a discounted factor balancing the importance between the immediate reward and the future rewards. In this problem formulation, $\gamma = 1$ which means the immediate reward has the same importance as the future rewards.

The objective of this REM problem is to find an optimal policy $\boldsymbol{\pi}^*$ such that the sum of future rewards is maximized, i.e.,

$$Q^*(s, a) = \max_{\boldsymbol{\pi}} Q_{\boldsymbol{\pi}}(s, a), \tag{4}$$

where $Q^*(s, a)$ is called optimal action-value function. With this optimal policy $\boldsymbol{\pi}^*$, we can derive the sequence of optimal actions $a^* = \text{argmax}_{a \in \mathcal{A}} Q^*(s, a)$ to maximize the sum of future rewards as Eq. (5), which is equivalent to minimize the total electricity costs.

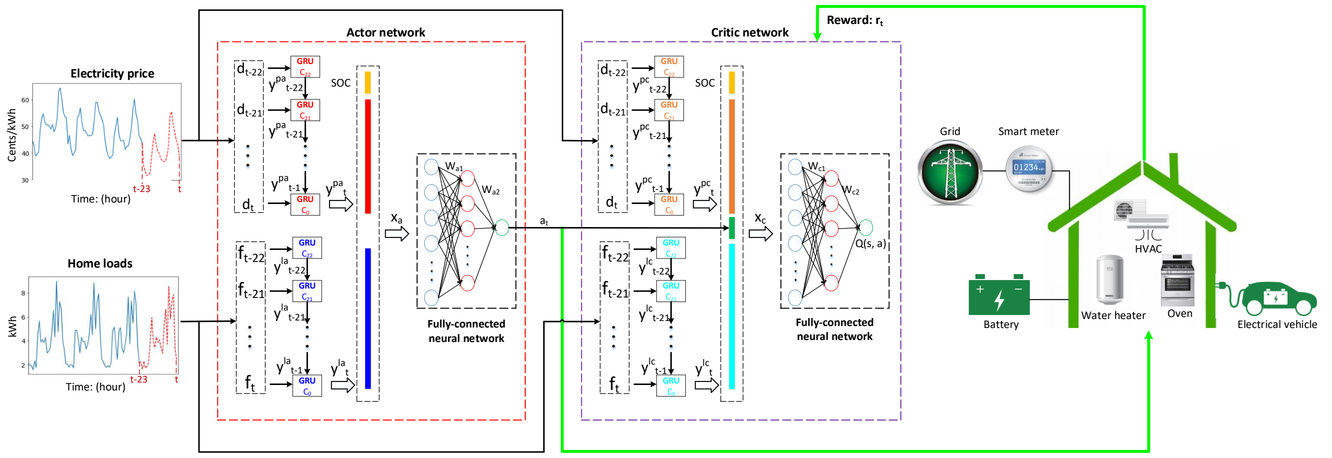


Fig. 2. The overall diagram of the deep RL based residential energy management. The proposed model consists of an actor network and a critic network. Their inputs are the battery SOC, past 24-h electricity price, and past 24-h home loads. The real-time action a_t is generated by the actor network, and the reward r_t is fed into the critic network.

$$Q^*(s_1, a) = \max \sum_{k=0}^{23} r_{k+1} = \min \sum_{k=0}^{23} P_{k+1} * a_{k+1} \tag{5}$$

where s_1 is the initial state.

III. Proposed Approach

It is hard to analytically determine the optimal policy π^* because the future home loads and electricity price are generally unknown. In order to solve this problem, Q-learning can be used to update the action-value function $Q(s, a)$ according to the Bellman equation [26] as

$$Q_{i+1}(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q_i(s_{t+1}, a_{t+1}) \middle| s_t = s, a_t = a \right]. \tag{6}$$

where i represents the number of iteration. $Q(s, a)$ will converge to the optimal action-value function $Q^*(s, a)$ [27] when $i \to \infty$.

Fig. 2 illustrates the overall diagram of the deep RL based residential energy management (REM). The inputs of the proposed model are the battery SOC, past 24-h electricity price, and past 24-h home loads. Then, the real-time action a_t is generated, and the reward r_t is calculated.

A. Architecture of Proposed Network

The architecture of the proposed deep RL based model in Fig. 2 consists of an actor network and a critic network. The actor network generates the action a_t, i.e., the amount of electricity to be purchased from the utility. The critic network outputs the action-value $Q(s, a)$.

1) Actor Network: Extracting good features from the input electricity price and home loads is important for this REM problem. Recurrent neural network has achieved great success in processing time-series data [28]–[30]. Gated recurrent unit (GRU) network [31], a type of recurrent neural network, is used to extract discriminative features from the input time-series data. The gradient of the past 24-h electricity price, $d_t = P_t - P_{t-1}, ..., d_{t-22} = P_{t-22} - P_{t-23}$, is inputted into the 23 cells of the GRU network, respectively. All the 23 cells share a set of parameters. For the cell C_{22}, its input is d_{t-22}, and its output y^{pa}_{t-22} is directly passed to the next cell C_{21}. Then, the past information y^{pa}_{t-22} and new input information d_{t-21} are fused in the cell C_{21}. This fusion process is repeated for the remaining cells.

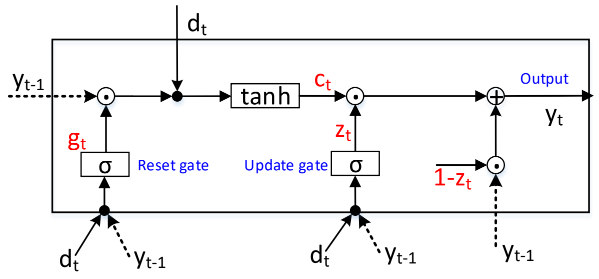


Fig. 3. Information flow in the GRU cell C_0 [31].

The architecture of the GRU cell C_0 is presented in Fig. 3. The reset gate g_t determines the amount of past information y_{t-1} to be stored into the cell. The reset gate is computed by

$$g_t = \sigma\left(W_g * d_t + U_g * y_{t-1} + b_g\right) \tag{7}$$

where σ denotes the sigmoid activation function, and W_g, U_g, and b_g represent the corresponding weights and bias term. The processed past information y_{t-1} is fused with the input information d_t as

$$c_t = \tanh\left[W * d_t + U * (g_t \odot y_{t-1})\right] \tag{8}$$

Algorithm 1 Training Process

```
1: Randomly initialize the main actor network $A(s \mid \theta^A)$ and critic network $Q(s, a \mid \theta^Q)$ with parameters $\theta^A$ and $\theta^Q$.
2: Initialize the parameters of the target actor network $A^{'}$ and critic network $Q^{'}$ as $\theta^{A^{'}} \leftarrow \theta^A$ and $\theta^{Q^{'}} \leftarrow \theta^Q$.
3: Initialize the replay buffer $\mathcal{D}$.
4: for Episode=1:M do
5:     Initialize state $s_1$.
6:     Initialize a random process $\mathcal{N}$ for action exploration.
7:     for Time step t=1:T do
8:         if Episode $<$ G then
9:             Randomly sample the action $a_t$ from an uniform distribution $\mathcal{U}(-e_{max}, e_{max})$.
10:            Execute the selected action $a_t$. Then, observe reward $r_t$ and transit to the new state $s_{t+1}$.
11:            Store transition $(s_t, a_t, r_t, s_{t+1})$ in the replay buffer $\mathcal{D}$.
12:        else
13:            Select the action $a_t = A(s_t \mid \theta^A) + \nu * \mathcal{N}_t$ according to the main actor network and the exploration noise.
14:            Execute the selected action $a_t$. Then, observe reward $r_t$ and transit to the new state $s_{t+1}$.
15:            Store transition $(s_t, a_t, r_t, s_{t+1})$ in the replay buffer $\mathcal{D}$.
16:            Randomly sample a minibatch of transitions $\mathcal{F} = \{(s_j, a_j, r_j, s_{j+1})\}_{j=1}^{\#\mathcal{F}}$ from $\mathcal{D}$.
17:            Calculate the target action-value $y_j = r_j + \gamma Q^{'}\left(s_{j+1}, A^{'}\left(s_{j+1} \mid \theta^{A^{'}}\right) \mid \theta^{Q^{'}}\right)$.
18:            Update the main critic network by minimizing the loss function $L_c = \sum_{j=1}^{\#\mathcal{F}} \left[y_j - Q\left(s_j, a_j \mid \theta^Q\right)\right]^2$.
19:            Update the main actor network with the sampled policy gradient:
               $\nabla_{\theta^A} J \approx \sum_{j=1}^{\#\mathcal{F}} \nabla_a Q\left(s, a \mid \theta^Q\right) |_{s=s_j, a=A(s_j)} \nabla_{\theta^A} A\left(s \mid \theta^A\right) |_{s_j}$.
20:            Update the target networks as:
               $\theta^{Q^{'}} \longleftarrow \tau\theta^Q + (1-\tau)\theta^{Q^{'}}$
               $\theta^{A^{'}} \longleftarrow \tau\theta^A + (1-\tau)\theta^{A^{'}}$
21:        end if
22:    end for
23: end for
```

where $\tanh$ is the hyperbolic tangent activation function, $\odot$ denotes the element-wise multiplication operation, and W, U, and b denote the weights and bias term. Then, the update gate z_t decides the amount of information to be outputted from c_t and y_{t-1}. Similar to the reset gate, the update gate is computed as

$$z_t = \sigma\left(W_z * d_t + U_z * y_{t-1} + b_z\right) \tag{9}$$

where W_z, U_z, and b_z are weights and bias term. Thus, the final output y_t is

$$y_t = z_t \odot c_t + (1 - z_t) \odot y_{t-1} \tag{10}$$

The output of the two GRU networks are concatenated with the battery SOC. These concatenated features x_a are inputted into a three-layer fully-connected neural network. The value of the hidden unit is computed as

$$h_a = relu\left(W_{a1} * x_a + b_{a1}\right) \tag{11}$$

where $relu$ denotes the rectified linear activation function, and W_{a1} and b_{a1} are the weights and bias term. Then, the output of this fully-connected neural network is calculated by

$$o = \sigma\left(W_{a2} * h_a + b_{a2}\right) \tag{12}$$

Then, o is scaled to the range of $-e_{max} \sim e_{max}$ and outputted as a_t.

2) Critic Network: Similar to the actor network, the features of the electricity price and the home loads are extracted by two GRU networks, respectively. These features are concatenated with the output of the actor network and the battery SOC. Then, the concatenated features are fed into the three-layer fully-connected neural network. Its output is the action-value $Q(s, a)$.

B. Training Algorithm

The proposed model is trained to generate optimal actions based on the deep deterministic policy gradient [32]. The training process is provided in Algorithm 1. Apart from the main actor network and critic network, two target networks are applied to calculate the target action-value. Its initial parameters are copied from the main networks. When episode is smaller than G, we refer this phase as pretrain phase where the action a_t is randomly sampled from an uniform distribution $\mathcal{U}(-e_{max}, e_{max})$. After the pretrain phase, the action a_t is computed according to the main actor network and the exploration noise as

$$a_t = A(s_t \mid \theta^A) + \nu * \mathcal{N}_t. \tag{13}$$

where the noise $\mathcal{N}_t$ is sampled from an uniform distribution bounded between 0 and 1. The noise scale ν is calculated as

$$\nu = 0.2 * e_{max} + \nu_0 * \beta^{e-e_p} \tag{14}$$

where $\nu_0 = 1.8 * e_{max}$, β denotes the noise decay rate, e represents the number of episode, and e_p indicates the number of pretrain episodes. The noise scale decays during the training process. After executing the selected action, the reward r_t is observed, and the environment advances to the next state s_{t+1}. The transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer $\mathcal{D}$. Then, a minibatch of transitions $\mathcal{F} = \{(s_j, a_j, r_j, s_{j+1})\}_{j=1}^{\#\mathcal{F}}$ is randomly sampled from $\mathcal{D}$. With these transitions, the parameters of the critic network is updated by minimizing the loss function

$$L_c = \sum_{j=1}^{\#\mathcal{F}} \left[y_j - Q\left(s_j, a_j \mid \theta^Q\right)\right]^2 \tag{15}$$

where $y_j = r_j + \gamma Q^{'}\left(s_{j+1}, A^{'}\left(s_{j+1} \mid \theta^{A^{'}}\right) \mid \theta^{Q^{'}}\right)$ is the target action-value calculated by the target actor network $A^{'}$ and the target critic network $Q^{'}$. The parameters of the actor network are updated with the sampled policy gradient

$$\nabla_{\theta^A} J \approx \sum_{j=1}^{\#\mathcal{F}} \nabla_a Q\left(s, a \mid \theta^Q\right)|_{s=s_j, a=A(s_j)} \nabla_{\theta^A} A\left(s \mid \theta^A\right)|_{s_j} . \tag{16}$$

Then, the parameters of the actor network are updated by

$$\theta^A \longleftarrow \theta^A + \alpha \nabla_{\theta^A} J \tag{17}$$

where α denotes the learning rate. Then, the parameters of the target networks are updated by slowly tracking the main networks

$$\begin{aligned} \theta^{Q^{'}} &\longleftarrow \tau\theta^Q + (1-\tau)\theta^{Q^{'}} \\ \theta^{A^{'}} &\longleftarrow \tau\theta^A + (1-\tau)\theta^{A^{'}} \end{aligned} \tag{18}$$

where τ is a small constant coefficient.

IV. Experimental Results

The performance of the proposed deep RL based REM is evaluated by simulation analysis. The simulation setup is introduced in Section IV-A. Then, the training results are provided in Section IV-B. Finally, in Section IV-C, the proposed model is evaluated and compared with several benchmark algorithms.

A. Simulation Setup

1) The Environment: The simulation is based on the real-world REM scenario, as shown in Fig. 1. The REM system decides how much electricity to be purchased from the utility. The allowed maximum purchased electricity $e_{max} = 10kWh$. The purchased electricity can be used to supply the home demand and charge the battery. The battery can discharge to power the home loads as well. The battery capacity is 20 kWh and the allowed minimum battery energy $E_{min} = 2kWh$. The penalty factor $\lambda = 100$. The electricity price is obtained from real-world electricity market, California ISO [25]. The price data starts from March 1st, 2014 and contains 300 days, where 200 days are used for training and the remaining 100 days

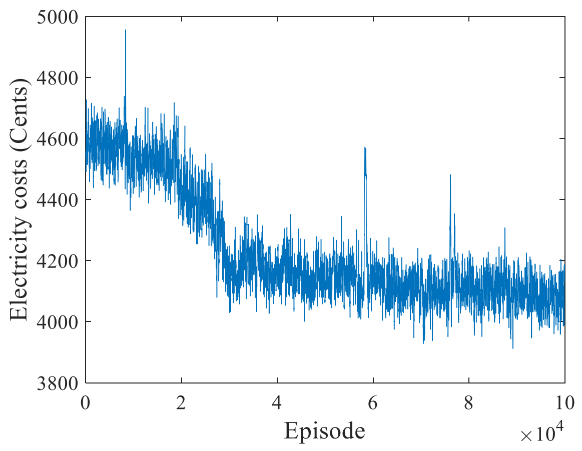


Fig. 4. The training process of the proposed model.

are used for testing. The household consumption is generated based on user behavior modeling in accordance with [7].

2) Model Structure: The structure of the proposed model is presented as follows. In the actor network, the GRU network has 23 cells, and its output is a 16-dimensional vector. The input layer of the fully-connected neural network has 33 units, and its hidden layer has 16 units. Its output is scaled to the range of $-10 \sim 10$. In the critic network, the GRU network has the same structure as the one in the actor network. The input layer of the fully-connected neural network has 34 units, and its hidden layer has 16 units.

3) Training Parameters: During the training process, the pretrain phase lasts 8000 episodes. The size of the replay buffer is set as 300,000. The size of the mini-batch is 64. The initial noise scale is $2 * e_{max} = 20kWh$ and the noise decay rate β is set as 0.9985. For the target network, the updating coefficient τ is 0.01.

B. Training Phase

The proposed model is trained for 100,000 episodes to generate the optimal actions. Each episode has 24 time steps. The training process of the proposed approach is presented in Fig. 4. During the first 8,000 episodes, the action is randomly sampled from the uniform distribution $\mathcal{U}(-e_{max}, e_{max})$. After that, the action is selected according to the main actor network and the exploration noise. Fig. 4 shows that the electricity costs decreases steadily during the training process. After about 60,000 episodes, the electricity costs converges around 4,000 cents. This training curve shows that the proposed model learns to minimize the electricity costs after sufficient training episodes.

C. Performance Evaluation

The performance of the proposed model is evaluated on the test days. In order to demonstrate the effectiveness of the proposed model, its performance is compared with some

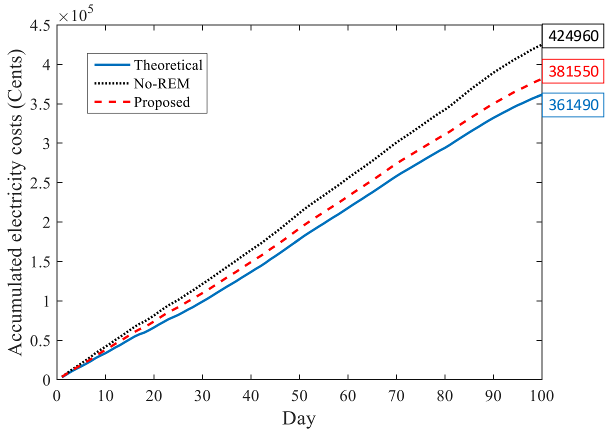


Fig. 5. The accumulated electricity costs of the proposed model and benchmark methods over the 100 test days.

benchmarks, including no-REM system, and theoretical optimum energy management. For the no-REM system, the home loads are directly powered by the utility and no DR policy is used. For the theoretical optimum method, we assume the future electricity price and home loads are all available. Thus, this problem becomes a deterministic optimization problem. Then, this problem is solved by an optimization algorithm, YALMIP [33]. It is worth noting that this method establishes a theoretically minimal daily electricity cost. However, it is hard to find this theoretical minimum since the future electricity price and home loads are generally unknown in practice.

The accumulated electricity costs of the proposed model and the benchmark methods over the 100 test days are shown in Fig. 5. The result of the proposed model is indicated by red dashed line while the results of the theoretical optimum method and no-REM system are represented by a blue solid line and a black dotted line. The accumulated electricity costs of the proposed model is reduced by 11.38 % compared to the case without REM. In addition, the result of the proposed model is close to the theoretical optimal one. These results verify the effectiveness of the proposed model in reducing the electricity costs.

In order to further evaluate the proposed model, the REM schedules over four consecutive days are illustrated in Fig. 6. Fig. 6a illustrates the hourly home loads (blue line) and electricity price (orange line). The battery energy over these four consecutive days is presented in Fig. 6b. These results show that the battery is charged when home loads and electricity price are low. On the contrary, the battery is discharged when home loads and electricity price are high. These results demonstrate the effectiveness of the proposed model.

V. Conclusion and Future Work

In this paper, the REM problem is formulated as a finite Markov decision process. The uncertainty of the real-time electricity price and the home loads are taken into account. A deep reinforcement learning based approach is proposed to find the optimal energy management strategy. The proposed approach does not require any knowledge about the uncertainty in home loads and electricity price. The proposed model consists of two networks, an actor network and a critic network. The actor network generates the action, i.e., the purchased electricity. The critic network assesses the performance of executing this action. The simulation results show that the proposed approach can greatly reduce the electricity costs compared to the case without REM. In the future work, we will consider a more complex scenario where we can shift the power consumption of the controllable loads such as water heater, HVAC, and electric vehicle. We will also compare with other traditional optimization methods.

Acknowledgment

This work was supported in part by National Science Foundation (NSF) under grant ECCS 1731672 and CCF 1439011.

References

[1] U. E. I. Administration, “International energy outlook 2017.” [Online]. Available: https://www.eia.gov/outlooks/ieo/

[2] D. Zhang, S. Li, M. Sun, and Z. ONeill, “An optimal and learning-based demand response and home energy management system,” *IEEE Transactions on Smart Grid*, vol. 7, no. 4, pp. 1790–1801, July 2016.

[3] O. Erdin, A. Tackaraolu, N. G. Paterakis, Y. Eren, and J. P. S. Catalo, “End-user comfort oriented day-ahead planning for responsive residential hvac demand aggregation considering weather forecasts,” *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 362–372, Jan 2017.

[4] S. A. Pourmousavi, S. N. Patrick, and M. H. Nehrir, “Real-time demand response through aggregate electric water heaters for load shifting and balancing wind generation,” *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 769–778, March 2014.

[5] S. Althaher, P. Mancarella, and J. Mutale, “Automated demand response from home energy management system under dynamic pricing and power and comfort constraints,” *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1874–1883, July 2015.

[6] A. Anvari-Moghaddam, H. Monsef, and A. Rahimi-Kian, “Optimal smart home energy management considering energy saving and a comfortable lifestyle,” *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 324–332, Jan 2015.

[7] N. G. Paterakis, O. Erdin, A. G. Bakirtzis, and J. P. S. Catalo, “Optimal household appliances scheduling under day-ahead pricing and load-shaping demand response strategies,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 6, pp. 1509–1519, Dec 2015.

[8] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. Slootweg, “On-line building energy optimization using deep reinforcement learning,” *arXiv preprint arXiv:1707.05878*, 2017.

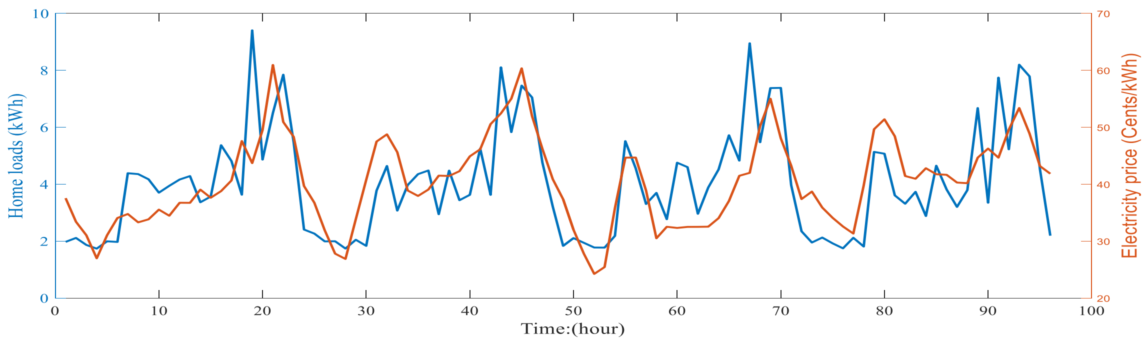
[9] M. Abdulgader, S. Lakshminarayanan, and D. Kaur, “Efficient energy management for smart homes with grey wolf optimizer,” in *2017 IEEE International Conference on Electro Information Technology (EIT)*, May 2017, pp. 388–393.

[10] F. Y. Melhem, O. Grunder, Z. Hammoudan, and N. Moubayed, “Optimization and energy management in smart home considering photovoltaic, wind, and battery storage system with integration of electric vehicles,” *Canadian Journal of Electrical and Computer Engineering*, vol. 40, no. 2, pp. 128–138, Spring 2017.

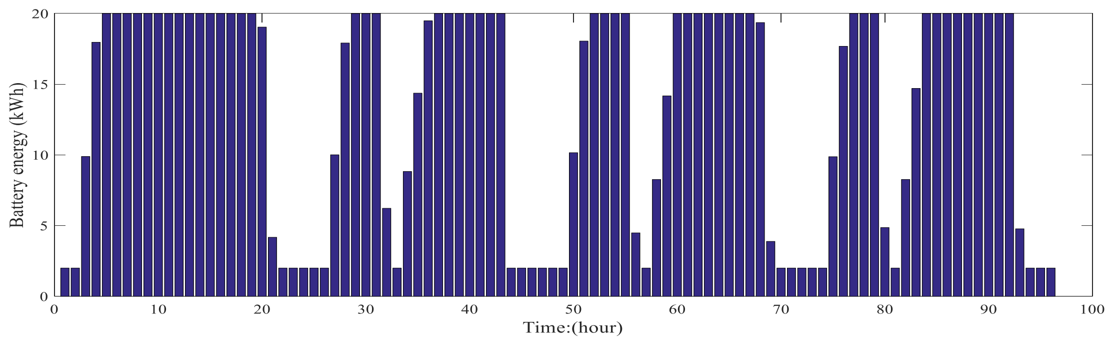
[11] Q. Wei, F. L. Lewis, G. Shi, and R. Song, “Error-tolerant iterative adaptive dynamic programming for optimal renewable home energy scheduling and battery management,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 12, pp. 9527–9537, Dec 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[13] Z. Wan and H. He, “Weakly supervised object localization with deep convolutional neural network based on spatial pyramid saliency map,” in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 4177–4181.



(a) Hourly home loads (blue line) and hourly electricity price (orange line).



(b) Battery energy (blue bar)

Fig. 6. REM schedules over four consecutive days: (a) Hourly home loads (blue line) and hourly electricity price (orange line); (b) Battery energy (blue bar). The battery is charged when home loads and electricity price are low. On the contrary, the battery is discharged when home loads and electricity price are high.

[14] L. Ma, Z. Lu, and H. Li, “Learning to answer questions from image using convolutional neural network,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[15] Z. Wan, H. He, and B. Tang, “A generative model for sparse hyperparameter determination,” *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2017.

[16] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[17] Z. Wan, X. Hu, H. He, and Y. Guo, “A learning based approach for social force model parameter estimation,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 4058–4064.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[20] J. Heinrich and D. Silver, “Deep reinforcement learning from self-play in imperfect-information games,” *arXiv preprint arXiv:1603.01121*, 2016.

[21] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, “Deepstack: Expert-level artificial intelligence in heads-up no-limit poker,” *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[22] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, “Optimal and autonomous control using reinforcement learning: A survey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–21, 2017.

[23] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, “Deep direct reinforcement learning for financial signal representation and trading,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, March 2017.

[24] M. Boaro, D. Fuselli, F. D. Angelis, D. Liu, Q. Wei, and F. Piazza, “Adaptive dynamic programming algorithm for renewable energy scheduling and battery management,” *Cognitive Computation*, vol. 5, no. 2, pp. 264–277, Jun 2013.

[25] C. ISO, “http://oasis.caiso.com/mrioasis/logon.do.”

[26] R. Bellman, “Dynamic programming,” 1958.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[28] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.

[29] M. Han and M. Xu, “Laplacian echo state network for multivariate time series prediction,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 1, pp. 238–244, Jan 2018.

[30] E. Santana, M. S. Emigh, P. Zegers, and J. C. Principe, “Exploiting spatio-temporal structure with recurrent winner-take-all networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–9, 2017.

[31] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[33] J. Löfberg, “Yalmip : A toolbox for modeling and optimization in matlab,” in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.