Demand-Side Scheduling Based on Multi-Agent Deep Actor-Critic Learning for Smart Grids

Joash Lee[1], Wenbo Wang[2], Dusit Niyato[2]
[1]*Energy Research Institute @ NTU, Nanyang Technological University, Singapore*
[2] *School of Computer Science and Engineering, Nanyang Technological University, Singapore*
{l.joash, wbwang, dniyato}@ntu.edu.sg

***Abstract*—We consider the problem of demand-side energy management, where each household is equipped with a smart meter that is able to schedule home appliances online. The goal is to minimize the overall cost under a real-time pricing scheme. While previous works have introduced centralized approaches in which the scheduling algorithm has full observability, we propose the formulation of a smart grid environment as a Markov game. Each household is a decentralized agent with partial observability, which allows scalability and privacy-preservation in a realistic setting. The grid operator produces a price signal that varies with the energy demand. We propose an extension to a multi-agent, deep actor-critic algorithm to address partial observability and the perceived non-stationarity of the environment from the agent's viewpoint. This algorithm learns a centralized critic that coordinates training of decentralized agents. Our approach thus uses centralized learning but decentralized execution. Simulation results show that our online deep reinforcement learning method can reduce both the peak-to-average ratio of total energy consumed and the cost of electricity for all households based purely on instantaneous observations and a price signal.**

***Index Terms*—Reinforcement learning, smart grid, deep learning, multi-agent systems, task scheduling**

I. Introduction

In the past decade, Demand-Side Management (DSM) with price-based demand response has been widely considered to be an efficient method of incentivizing users to collectively achieve better grid welfare [1]. In a typical DSM scenario, an independent utility operator manipulates the aggregate energy demand by adopting a time-varying pricing scheme that is announced to users on a rolling basis. In response to the pricing scheme, the users voluntarily schedule or shift their appliance loads with the aim of minimizing their individual costs (see the example in [2]). Typical objectives can be to minimize energy consumption, costs, peak load demand [2]–[8] or maximize the utility of certain actors [9]–[11].

This research is supported by the National Research Foundation (NRF), Singapore, under Singapore Energy Market Authority (EMA), Energy Resilience, NRF2017EWT-EP003-041, Singapore NRF2015-NRF-ISF001-2277, Singapore NRF National Satellite of Excellence, Design Science and Technology for Secure Critical Infrastructure NSoE DeST-SCI2019-0007, A*STAR-NTU-SUTD Joint Research Grant on Artificial Intelligence for the Future of Manufacturing RGANS1906, Wallenberg AI, Autonomous Systems and Software Program and Nanyang Technological University (WASP/NTU) under grant M4082187 (4080), Singapore Ministry of Education (MOE) Tier 1 (RG16/20), and NTU-WeBank JRI (NWJ-2020-004), Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI).

Under the framework of automatic price-demand scheduling, a variety of problem formulations have been previously proposed. The DSM problem can be formulated under a centralized or decentralized framework. It can also be categorized under a planning-based framework or an online framework that allow users to react to time-variant features. Examples of centralized planning include casting DSM as an optimization problem [7], [9], while an example of a centralized online approach is the scheduling problem proposed by [8]. However, a shortcoming of these methods is their inability to account for unknown parameters or stochasticity in the DSM system such as random task arrivals and real-time pricing. An alternative method is to model the system dynamics as a Markov Decision Process (MDP) with partially observability [12]. Instead of planning on the premise of a known system model, strategies can be learned through reinforcement learning [3], [4].

In this paper, we propose a Markov game-based framework that allows household energy users within a microgrid to manage their energy consumption both decentrally and online via their smart meters. Our second contribution is an extension of a deep actor-critic algorithm to train these smart meter-enabled household agents to minimize the cost of energy under real-time pricing. (We shall also refer to households as 'agents' in the reinforcement learning context from hereon.) Our proposed approach offers two main advantages. The main advantage of decentrally-controlled energy-managing smart meters is that, after the training phase, each household agent is able to act independently in the execution phase, without requiring specific information about other households — each household agent will not be able to observe the number or nature of devices operated by other households, the amount of energy they individually consume, and when they consume this energy. This allows for privacy-preservation and scalability in the execution phase. The main advantage of the online strategy-learning framework is that each household agent can respond appropriately to fluctuations brought about by stochastic processes within the microgrid system. We introduce details of the system model in Section III.

Our aforementioned aim addresses two key challenges. Firstly, the model of the microgrid system and its pricing methodology is not available to each household. Secondly, the limitation in the observation available to each household and the lack of visibility regarding the strategy of other households creates the local impression of a non-stationary environment.

To overcome the first mentioned key challenge, we propose

a model-free policy-based reinforcement learning method in Section IV for each household to learn their optimal DSM strategies. To address the second key challenge, we propose a centralized critic network which provides information to guide the learning processes of the households. Experiments are presented in Section V. By comparing our method to an offline method that plans consumption based on a predicted price schedule [7], we show that our method is better able to handle the uncertainties associated with real-time pricing.

II. Related Work and Preliminaries

A number of previous works have applied reinforcement learning to train decision-making entities in smart grid settings [5], [10], [11], [13]. These studies applied tabular value-based reinforcement learning algorithms to optimize energy costs. Consequently, these methods were limited to processing only discretized state and action spaces.

Recently, there have also been studies that have introduced methods utilizing deep neural networks for multi-agent reinforcement learning, albeit to different classes of problems [14]–[16]. These newly introduced methods are extensions of the now-canonical deep Q-network (DQN) algorithm for the multi-agent environment: they utilize a central value-function approximator to share information. However, as explained in section IV, a value-based reinforcement learning algorithm would not suit the problem we introduce in this paper because we consider a reward function that cannot be fully described by only the current state and action.

In this paper, we propose a policy-based algorithm based on the Proximal Policy Optimization (PPO) algorithm [17] for a multi-agent framework. We consider the partially observable Markov game framework [18], which is an extension of the Markov Decision Process (MDP) for multi-agent systems. A Markov game for N agents can be mathematically described by a tuple $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_n\}_{n\in\mathcal{N}}, \{\mathcal{O}_n\}_{n\in\mathcal{N}}, \mathcal{T}, \{r_n\}_{n\in\mathcal{N}} \rangle$, where $\mathcal{N}$ is the set of individual agents, n is the agent index, and $\mathcal{S}$ describes set of all possible overall states of all agents in the environment.

At each discrete time step k, each agent n is able to make an observation $o_n \in \mathcal{O}_n$ of its own state. Based on this observation, each agent takes an action that is chosen using a stochastic policy $\pi_{\theta_n} : \mathcal{O}_n \mapsto \mathcal{A}_n$, where $\mathcal{A}_n$ is the set of all possible actions for agent index n. The environment then determines each agent's reward as a function of the state and each agent's action, such that $r_n(k) : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \mapsto \mathbb{R}$. It moves on to the next state according to its state transition model $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \to \mathcal{S}'$.

III. System Model

We consider a DSM problem in a microgrid of N households that each have M household appliances. Each household is equipped with a smart meter that is capable of controlling all household appliances. The problem that we present is a more advanced form of the demand-side scheduling problem than that presented in [7] where each household has no advanced knowledge of upcoming tasks. We primarily consider interactive background appliances. These appliances are actively turned on by the household, but the actual time of operation poses a less immediate impact to the user. The appliances that we consider in each household can be, for example, washing machines, clothes dryers, storage water heaters, dish washers or refrigerators.

Each household's local operational state can be expressed as $\mathbf{s}_n = [\mathbf{x}_n^\top, \mathbf{t}_n^\top, \mathbf{l}_n^\top, \mathbf{q}_n^\top]^\top$. It consists of an indicator of whether each household appliance (super-scripted index) is turned on or off $\mathbf{x}_n = [x_n^1, \ldots, x_n^M]^\top$, the length of time before the next task can commence $\mathbf{t}_n = [t_n^1, \ldots, t_n^M]^\top$, the number of tasks queued for each appliance $\mathbf{q}_n = [q_n^1, \ldots, q_n^M]^\top$, and the lengths of the subsequent tasks each appliance has in queue $\mathbf{l}_n = [l_n^1, \ldots, l_n^M]^\top$. The overall state is simply the joint operational state of all the users: $\mathbf{s} = [\mathbf{s}_1, \ldots, \mathbf{s}_N]$. We adopt a practical assumption that each household can only observe its own internal state along with the published price at the previous time step $p(k-1)$, such that $\mathbf{o}_n(k) = [\mathbf{s}_n(k)^\top, p(k-1)]^\top$.

The task arrival for each appliance corresponds to the household's demand to turn it on, and is based on a four-week period of actual household data from the Smart* dataset [19]. We first discretise each day into a set of H time intervals $\mathcal{H} = \{0, 1, 2, ..., H-1\}$ of length T, where each interval corresponds to the time in the simulated system. The simulation time interval for each time-step in the Markov game is determined with the following relation:

$$h(k) = \mod(k, H). \tag{1}$$

Next, we count the number of events in each interval where the appliances of interest were turned on. This is used to compute an estimate of the probability that each appliance receives a new task during each time interval, $\Pr({q'}_n^m = 1 | h(k))$. The task arrival for each appliance ${q'}_n^m$ at each time interval k is thus modelled with a Bernoulli distribution. Each task arrival ${q'}_n^m$ for a particular appliance m in household n is added to the queue q_n^m.

The duration of the next task to be executed is denoted by l_n^m. Its value is updated by randomly sampling from an exponential distribution whenever a new task is generated (i.e. ${q'}_n^m = 1$). The rate parameter λ_n^m for the distribution for each appliance is assigned to be the reciprocal of the approximate average duration of tasks in the above-mentioned four-week period selected from the Smart* dataset [19]. We choose to keep the task duration constrained to $l \in [T, \infty]$.

The use of the described probabilistic models for generation of task arrivals and task lengths enable us to capture the variation in household energy demand throughout the course of an average day while preserving stochasticity in energy demand at each household.

We consider that each household's scheduler is synchronized to operate with the same discrete time steps of length T. At the start of a time step, should there be a task that has newly arrived into an empty queue ($q_n^m = 1$), the naive strategy of inaction would have the task start immediately such that $t_n^m = 0$. This corresponds with the conventional day-to-day scenario where a household appliance turns on immediately when the user demands so. However, in this paper we consider a setting where each household has an intelligent scheduler

that is capable of acting by delaying each appliance's starting time by a continuous time period $a_n^m \in [0, T]$. The joint action of all users is thus: $\mathbf{a} = [\mathbf{a}_1, \ldots, \mathbf{a}_N]^\top$.

Once an appliance has started executing a task from its queue, q_n^m is shortened accordingly. We consider that the appliance is uninterruptible and operates with a constant power consumption P_n^m. If an appliance is in operation for any length of time during a given time step, we consider its operational state to be $x_n^m = 1$. Otherwise, $x_n^m = 0$.

Let $d_n^m(k)$ denote the length of time that an appliance m is in operation during time step k for household n. The energy consumption of a household $E_n(k)$ during time step k is thus:

$$E_n(k) = \sum_{m=1}^{M} d_n^m(k) P_n^m. \tag{2}$$

On the grid side, we consider a dynamic energy price $p(k)$ that is a linear function of the Peak-Average-Ratio [1], [7] of the energy consumption in the previous κ time steps:

$$p(k) = \frac{\kappa T \max\limits_{k-\kappa+1 \leq k} \sum\limits_{n=1}^{N} E_n(k)}{\sum\limits_{n=1}^{N} E_n(k)}. \tag{3}$$

Each household receives a private reward signal that is a weighted sum of a cost objective and a soft constraint:

$$r_n(k) = -r_{c,n}(k) + w r_{e,n}(k). \tag{4}$$

The first component of the reward function $r_{c,n}$ is the monetary cost incurred by each household at the end of each time step. The negative sign of the term (5) in (4) encourages the policy to delay energy consumption until the price is sufficiently low.

$$r_{c,n}(k) = p(k) \times E_n(k) \tag{5}$$

The second component of the reward is a soft constraint $r_{e,n}$ tunable by weight w. The soft constraint encourages the policy to schedule household appliances at a rate that matches the average rate of task arrival into the queue.

$$r_{e,n}(k) = E_n(k) \tag{6}$$

The state transition for all households can be considered to be Markovian because the transition $\mathcal{T}$ to the next overall system state is dependent on the current state and the actions of all household agents. In contrast, we recall from (3) and (4) that the price of energy, and consequently each agent's reward, is a function of not only the current state and actions, but also of the history of previous states. Our pricing-based energy demand management process is, therefore, a generalization of the Markov game formulation [18].

IV. Semi-Distributed Deep Reinforcement Learning for Task Scheduling

The DSM microgrid problem that we introduced in the previous section presents various constraints and difficulties in solving it: (i) each household can choose its action based only on its local observation and the last published price, (ii) the reward function is dependent on the states and actions beyond the last time step, (iii) the state and observation spaces are continuous, (iv) the household agents have no communication channels between them, and (v) a model of the environment is not available. Constraint (iv) is related to the general problem of environmental non-stationarity: concurrent training of all agents would cause the environmental dynamics to constantly change from the perspective of a single agent, thus violating Markov assumptions.

Requirements (i) and (ii) and environmental non-stationarity mean that value-based algorithms such as Q-learning are not suitable, because these algorithms depend on the Markov assumption that the state transition and reward functions are dependent only on the state and action of a single agent at the last time step. Furthermore, requirement (v) rules out model-based algorithms. Thus, in this section, we propose an extension of the actor-critic algorithm Proximal Policy Optimization (PPO) [17] for the multi-agent setting which we will refer to as Multi-Agent PPO (MAPPO).

Let $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$ be the set of policies, parameterized by $\theta = \{\theta_1, \theta_2, ..., \theta_N\}$, for each of the N agents in the microgrid. Each policy $\pi_n(\mathbf{a}_n|\mathbf{s}_n)$ is stochastic, in that it produces and samples from a probability distribution of actions given the observed state. The objective function that we seek to maximize is the expected sum of discounted reward $J(\theta) = E_{\mathbf{s}\sim Pr^\pi, \mathbf{a}\sim\pi_\theta}[\sum_{k=i}^{K} \gamma^{k-i} r_k]$, where Pr^π is the probability distribution of states under policy set π, and r_k is obtained based on (4). In canonical policy gradient algorithms such as A2C, gradient ascent is performed to improve the policy:

$$\begin{aligned} &\nabla_{\theta_n} J_n(\theta_n) = \\ &E_{\mathbf{s}\sim Pr^\pi, \mathbf{a}_n\sim\pi_n} \left[\nabla_{\theta_n} \log \pi_n(\mathbf{a}_n|\mathbf{s}_n) Q_n^{\pi_n}(\mathbf{s}_n, \mathbf{a}_n)\right]. \end{aligned} \tag{7}$$

This policy gradient estimate is known to have high variance. It has been shown that this problem is exacerbated in multi-agent settings [14]. For this reason, we propose the implementation of PPO as it limits the size of each policy update. Additionally, we augment the individually trained actor networks with a central critic network $\hat{V}^\pi$ that approximates the value function for each household V_n^π based on the overall system state, thus receiving information for the training of cooperative actions by individual households. The objective function we propose is:

$$\begin{aligned} L(\theta_n) = E_k\Big[\min\Big(&\rho_k(\theta_n)\hat{A}_n^\pi(k), \max\big(1-\epsilon, \\ &\min(\rho_k(\theta_n), 1+\epsilon)\big)\hat{A}_n^\pi(k)\Big) + \lambda H(\pi_n)\Big], \end{aligned} \tag{8}$$

where $\rho_k(\theta_n) = \frac{\pi_n(\mathbf{a}_k|\mathbf{o}_k)}{\pi_{n,old(\mathbf{a}_k|\mathbf{o}_k)}}$, and both ϵ and λ are hyperparameters. $\hat{A}_n^\pi(k)$ is the estimated advantage value, which is computed using the critic network:

$$\hat{A}_n^\pi(\mathbf{s}, \mathbf{a}_n) \approx r(\mathbf{s}, \mathbf{a}_n) + \gamma \hat{V}^\pi(\mathbf{s}(k+1))_n - \hat{V}^\pi(\mathbf{s}(k))_n. \tag{9}$$

We fit the critic network, parameterized by ϕ, with the following loss function using gradient descent:

$$L(\phi) = \sum_n \sum_k \Big(\hat{V}^\pi(\mathbf{s}(k))_n - \mathbf{y}_n(k)\Big)^2. \tag{10}$$

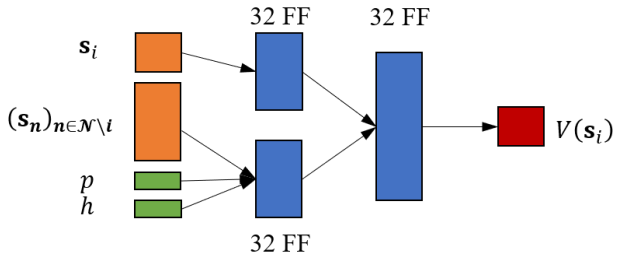

Fig. 1: The architecture of a centralized critic neural network.

TABLE I: Time-related hyper-parameters for simulations of the 32-household microgrid system

Symbol	Hyper-parameter	Value
T	Period of each time step	0.5 hours
-	Episode length	240 time steps
-	Time steps per iteration of the training algorithm	5040 time steps

where the target value $\mathbf{y}(k) = r_n(k)+\gamma\hat{V}^{\pi}(\mathbf{s}(k+1))_n$. As the notation suggests, this update is based on the assumption that the policies of the other agents, and hence their values, remain the same. In theory, the target value must be re-calculated every time the critic network is updated. However, in practice, we take a few gradient steps at each iteration of the algorithm.

The centralized critic-network utilizes a branched neural network architecture shown in Figure 1. To compute the estimated state value of a particular household i, V_i, we provide the following input vector to the centralized critic-network: $[\mathbf{s}_i^\top\ \mathbf{s}_1^\top\ \mathbf{s}_2^\top\ \ldots\ \mathbf{s}_1^\top\ \mathbf{s}_{i-1}^\top\ \mathbf{s}_{i+1}^\top\ \ldots\mathbf{s}_N^\top]^\top$. The critic network would thus provide an estimate of the quantity $V^{\pi}(\mathbf{s})_i = \sum_k E_{\pi_\theta}[r(\mathbf{s}_i,\mathbf{a}_i)|\mathbf{s}_1,\mathbf{s}_2,\ldots,\mathbf{s}_{i-1},\mathbf{s}_{i+1},\ldots,\mathbf{s}_N]$.

V. Experiments

In this section, we present the settings and environmental parameters that we use for experimentation, evaluate our proposed Multi-Agent PPO (MAPPO) algorithm against non-learning agents and the offline Energy Consumption Scheduler (ECS) method by [7], examine the effect of augmenting local observations in proposed system model, and consider the effects of household agents learning shared policies.

A. System Setup

We present experiments on a simulated system of identical households, each with 5 household appliances. As mentioned, their parameters relating to power consumption P_n^m, task arrival rates $p_n^m(k)$ and lengths of operation λ_n^m are chosen to be a loose match to data sampled from the Smart* 2017 dataset [19], and thus resemble typical household usage patterns.

Hyper-parameters relating to the simulation time scales are shown in Table I. Each experiment is run with a number of random seeds; the extent of their effect is represented by shaded areas on the graphs shown.

For this chosen experimental setup, we run two baseline performance benchmarks that were non-learning: (i) Firstly, an agent that assigns delay actions of zero for all time steps. This represents the default condition in where the household occupants dictate the exact time that all appliances turn on.

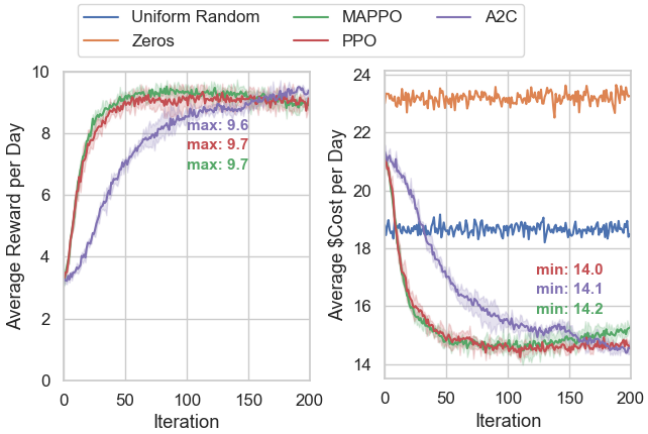

Fig. 2: Training progress for a 32-household microgrid system

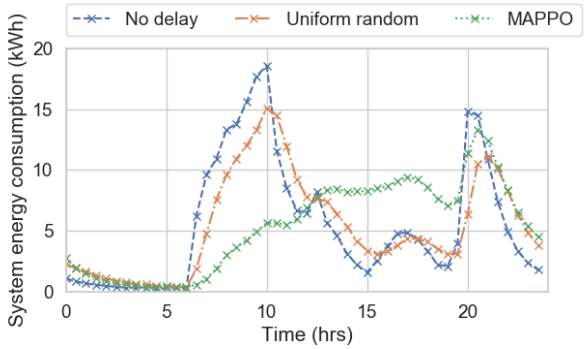

Fig. 3: Average aggregate energy consumed by 32-household microgrid for each time step of the day.

(ii) A second baseline agent randomly samples delay actions from a uniform distribution with range $[0, T]$.

The results from the training process are compared in Figure 2. The y-axes show the average reward gained (left), and the costs incurred (right) by all households for each day within each batch of samples. All trained policies demonstrate a sharp improvement in reward achieved from the onset of training. The monetary cost decreases with increasing reward, showing that the reward function is suitably chosen for the objective of minimizing cost.

In Figure 3, we plot the average total energy consumed by the entire microgrid system for each time step of the day. Note that the x-axis shows the time in the simulated environment. The blue line shows the energy-time consumption history if all home appliance were to be run immediately on demand; it reveals peaks in demand in the morning and evening, presumably before and after the homes' occupants are out to work, and lows in demand in the night time. The uniform-random-delay agent dampens these peaks in demand by shifting a number of tasks into the following time steps (note the lag in energy consumption compared to the zero-delay agent). In contrast, the agent trained using reinforcement learning is not only able to delay the morning peak demand, but also spread it out across the afternoon. The result is a lower PAR, and hence a lower monetary cost per unit energy.

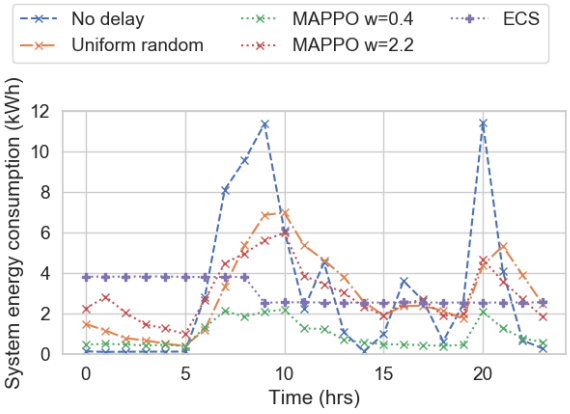

Fig. 4: Average aggregate energy consumed by 10-household microgrid for each time step of the day

Although the morning peak in energy demand is smoothed out by the MAPPO-trained households, the same effect is less pronounced for the evening peak. This can be attributed to the training procedure: the batch of experience gathered in each iteration of the algorithm began at 00:00 hrs, and terminated at the end of a 24 hour cycle, providing fewer opportunities for the agents to learn that the evening demand can be delayed to the early hours of the subsequent day. Moreover, delaying tasks on the last cycle of each batch would have led to a lower value of soft-constraint reward $r_{e,n}$, but an un-materialized cost improvement (higher $r_{c,n}$) for the subsequent morning (because of the midnight cut-off).

The results discussed above are significant for our particular application because it shows that while each household is controlled by different policies, all households are able to learn a cooperative strategy that flattens energy demand and consequently decrease the average cost for all users. The ability of the agents to learn in a decentralized manner is compatible with a microgrid system where there may be limited bandwidth to both receive and transmit information centrally. The individual user, or a small cluster of users, could improve their policy using local computing resources, while receiving only the output signal from the centralized critic network. Conversely, if computing resources for neural network training are only available centrally, update gradients can be transmitted to individual actor networks and result in similar performance.

B. Comparison with ECS

We compare our proposed MAPPO based method to the Energy Consumption Scheduler (ECS) method proposed by [7]. Instead of making decisions online as with MAPPO, ECS requires the following information in advance to schedule all energy consumption for the next day: (i) the mathematical function used to determine the price of electricity for each time step, and (ii) the total planned energy consumption for each household appliance for the entire day, computed as follows:

$$\sum_{h=0}^{H-1} E_n^m = \sum_{h=0}^{H-1} \left(\Pr(q'^m_n = 1|h) \times l_n^m \times P_n^m\right). \quad (11)$$

TABLE II: Average reward obtained per day in a 10-household microgrid

Non-learning		ECS	MAPPO	
Uniform Random	**Zeros**	**w=2.2**	**w=0.4**	**w=2.2**
-0.1	-13.8	7.0	-0.1	8.0

For a better comparison, we adapted the experimental setup to match the method used for ECS in its original paper: we considered a system of $N = 10$ households, split each day into $H = 24$ equal time intervals, fixed the length of operation of each task such that $l_n^m = 1/\lambda_n^m$, and used a quadratic price function (12). We also consider different values of the multiplier w for the cost constraint in the reward function (4).

$$p_n(k) = b(h) \times E_n(k)^2. \quad (12)$$

In Figure 4, we plot the average aggregate energy consumed by the microgrid for each time step. Once again, the agents trained using reinforcement learning are able to flatten the peaks in energy consumption by spreading demand across subsequent time steps. The MAPPO agents trained with a coefficient of $w = 2.2$ consume almost the same energy over the entire day as the naive agents with zero delay; this constitutes a shift in energy demand that results in a decrease in the cost of energy.

In comparison to the MAPPO agents, ECS produces a flatter energy demand profile. Because the ECS algorithm receives information in advance, it is able to schedule appliances to operate in the early hours of the day to exploit lower prices. However, since the energy consumption inputs received by ECS are average values, each household experiences a mismatch between its actual energy demand and the schedule produced by ECS. Consequently, we compute a reward according to (4), where the value $r_{c,n}$ is taken to be the actual energy demand that is fulfilled by the ECS-generated schedule. The results presented in Table II show that the MAPPO agents with $w = 2.2$ achieve an average daily reward that is more than 10% higher than that achieved by ECS.

The results reveal the MAPPO method's key advantage, in that it can be utilized in a fully online scenario without requiring the price function or the total daily consumption in advance. It is able to rely on instantaneous observations of the local state and the price signal to determine how tasks should be scheduled by delaying them. This online approach and reliance on mainly locally observable information is more consistent with the nature of domestic electricity use.

Another advantage of MAPPO is that, in addition to shifting the demand of energy, we are able to implement energy reduction by tuning the coefficient w. Figure 4 shows that the MAPPO agents trained with a smaller coefficient of $w = 0.4$ exhibit lower energy demand than both the MAPPO agents with $w = 2.2$ and the ECS method.

C. Comparison with Methods using Decentralized Critics

Figure 2 also compares our proposed MAPPO algorithm to a similar PPO-based algorithm with decentralized critics, and the more traditional Advantage Actor-Critic (A2C) [20]

algorithm. The number of learnable parameters across all the N decentralized critic networks was chosen to be similar to that of the centralized critic, on the assumption that this would result in a similar computation resource requirements. The results show that all the agents trained with deep reinforcement learning perform better than the non-learning agents.

Our proposed MAPPO algorithm achieves the most sample-efficient learning and best performance, while the performance of the decentralized PPO algorithm is close. This demonstrates that good performance can be achieved by learning in a completely decentralized manner. This would be useful in scenarios where the connectivity of household smart meters is limited, but local computing power can be utilized.

The A2C algorithm is also able to reach the same level of performance, although it learns less efficiently with the same number of samples. Experiments with higher learning rates showed instabilities that manifested in exploding gradients – this was in spite of policy-based algorithms such as A2C being generally known to be more stable than value-based algorithms. The reason is that, in our proposed microgrid environment, the updates to the policies for other households under A2C have a large effect on the overall system dynamics, possibly worsening the non-stationarity of the environment as perceived by each household. In contrast, the nature of the actor network update in the PPO algorithm limits the change in the actions taken by the updated policy, thus leading to stable performance increments.

VI. Conclusion

In this paper, we proposed and implemented a smart grid environment, as well as a multi-agent extension of a deep actor-critic algorithm for training decentralized household agents to schedule household appliances with the aim of minimizing the cost of energy under a real-time pricing scheme. Our approach allows for an online approach to household appliance scheduling that is privacy-preserving, requiring only the local observation and a price signal from the previous time step to act. A joint critic is learned to coordinate training centrally.

Our results show that our proposed algorithm is able to train agents that achieve a lower cost and flatter energy-time profile than non-learning agents. Our algorithm also achieves quicker learning than independent agents trained using the canonical Advantage Actor-Critic (A2C) algorithm. Crucially, its ability to respond to stochastic environmental parameters resulted in increased utility in comparison to an optimization-based planning method.

Future work can involve the extension of the smart grid environment to include more features and real-world complexities; we identify two possibilities. Firstly, the architecture of the microgrid can be expanded to include more energy sources, which may be operated by producers or households. Secondly, we may expand the overall grid to include more agents and more types of agents, such as energy producers and brokers.

References

[1] J. S. Vardakas, N. Zorba, and C. V. Verikoukis, "A survey on demand response programs in smart grids: Pricing methods and optimization algorithms," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 152–178, Firstquarter 2015.

[2] R. Deng, Z. Yang, M. Chow, and J. Chen, "A survey on demand response in smart grids: Mathematical models and approaches," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 570–582, Jun. 2015.

[3] F. Ruelens, B. J. Claessens, S. Vandael, B. D. Schutter, R. Babuška, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2149–2159, Sep. 2017.

[4] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2312–2324, Sep. 2015.

[5] B. Kim, Y. Zhang, M. v. d. Schaar, and J. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2187–2198, 2016.

[6] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3698–3708, 2019.

[7] A. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320–331, Dec. 2010.

[8] S. Barker, A. Mishra, D. Irwin, P. Shenoy, and J. Albrecht, "Smartcap: Flattening peak electricity demand in smart homes," in *2012 IEEE International Conference on Pervasive Computing and Communications*, 2012, pp. 67–75.

[9] X. Cao, J. Zhang, and H. V. Poor, "Joint energy procurement and demand response towards optimal deployment of renewables," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 4, pp. 657–672, Aug 2018.

[10] P. P. Reddy and M. M. Veloso, "Strategy learning for autonomous agents in smart grid markets," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[11] R. Lu, S. H. Hong, and X. Zhang, "A dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach," *Applied Energy*, vol. 220, pp. 220–230, 2018.

[12] T. M. Hansen, E. K. P. Chong, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, "A partially observable markov decision process approach to residential home energy management," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 1271–1281, March 2018.

[13] E. C. Kara, M. Berges, B. Krogh, and S. Kar, "Using smart devices for system-level management and control in the smart grid: A reinforcement learning framework," in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, 2012, pp. 85–90.

[14] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[15] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[16] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. PMLR, vol. 80, 2018, pp. 4295–4304.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, San Francisco (CA), 1994, pp. 157 – 163.

[19] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," in *KDD Workshop on Data Mining Applications in Sustainability (SustKDD 2012)*, 2012.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. PMLR, vol. 48, 20–22 Jun 2016, pp. 1928–1937.