A Cooperative Multi-Agent Deep Reinforcement Learning Framework for Real-Time Residential Load Scheduling*

Chi Zhang
Department of Computer Science
University of Southern California
Los Angeles, CA
zhan527@usc.edu

Sanmukh R. Kuppannagari
Ming Hsieh Department of Electrical
and Computer Engineering
University of Southern California
Los Angeles, CA
kuppanna@usc.edu

Chuanxiu Xiong
Department of Computer Science
University of Southern California
Los Angeles, CA
chuanxix@usc.edu

Rajgopal Kannan
US Army Research Lab-West
Playa Vista, CA
rajgopak@usc.edu

Viktor K. Prasanna
Ming Hsieh Department of Electrical
and Computer Engineering
University of Southern California
Los Angeles, CA
prasanna@usc.edu

ABSTRACT

Internet-of-Things (IoT) enabled monitoring and control capabilities are enabling increasing numbers of household users with controllable loads to actively participate in smart grid energy management. Realizing an efficient real-time energy management system that takes advantage of these developments requires novel techniques for managing the increased complexity of the control action space in resolving multiple challenges such as the uncertainty in energy prices and renewable energy output along with the need to satisfy physical grid constraints such as transformer capacity. Addressing these challenges, we develop a multi-household energy management framework for residential units connected to the same transformer and containing DERs such as PV, ESS and controllable loads. The goal of our framework is to schedule controllable household appliances and ESS such that the cost of procuring electricity from the utility over a horizon is minimized while physical grid constraints are satisfied at each scheduling step. Traditional energy management frameworks either perform global optimization to satisfy grid constraints but suffer from high computational complexity (for example Integer Program, Mixed Integer Programming frameworks and centralized reinforcement learning) or perform decentralized real-time energy management without satisfying global grid constraints (for example multi-agent reinforcement learning with no cooperation). In contrast, we propose a cooperative multi-agent reinforcement learning (MARL) framework that i) operates in real-time, and ii) performs explicit collaboration to satisfy global

grid constraints. The novelty in our framework is two fold. Firstly, our framework trains multiple independent learners (IL) for each household in parallel using historical data and performs real-time inferring of control actions using the most recent system state. Secondly, our framework contains a low complexity knapsack based cooperation agent which combines the outputs of ILs to minimize cost while satisfying grid constraints. Simulation results show that our cooperative MARL approach achieves significant cost improvement over centralized reinforcement learning and day-ahead planning baselines. Moreover, our approach strictly satisfies physical constraints with no apriori knowledge of system dynamics while the baseline approaches have occasional violations. We also measure the training and inference time by ranging the number of households from 1 to 25. Results show that our cooperative MARL approach scales best among various approaches.

CCS CONCEPTS

• Computing methodologies → Multi-agent planning; Multi-agent reinforcement learning.

KEYWORDS

multi-agent, deep reinforcement learning, smart home, real-time load scheduling, internet-of-things

ACM Reference Format:

Chi Zhang, Sanmukh R. Kuppannagari, Chuanxiu Xiong, Rajgopal Kannan, and Viktor K. Prasanna. 2019. A Cooperative Multi-Agent Deep Reinforcement Learning Framework for Real-Time Residential Load Scheduling. In *IoTDI '19: Internet of Things Design and Implementation, April 15–18, 2019, Montreal, QC, Canada*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3302505.3310069>

1 INTRODUCTION

Household adoption of Distributed Energy Resources (DER) such as PV (photovoltaic) and ESS (energy storage systems) has increased significantly in recent years. Solar generated power is expected to grow to 14% and 27% of the total power supply by 2030 and 2050, respectively [1]. Residences now come equipped with controllable

*This work has been funded by the Department of Energy (DoE) under award number DE-EE0008003 and the U.S. National Science Foundation under grant number 1637372.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IoTDI '19, April 15–18, 2019, Montreal, QC, Canada

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6283-2/19/04...\$15.00

<https://doi.org/10.1145/3302505.3310069>

loads including dishwashers, laundry and HVAC systems and EVs (Electric Vehicles) that can be used to participate in Demand Response programs – a classic approach for passive participation in grid management. Now, with 5G supported IoT technology and bidirectional AMI meters, real-time monitoring and control of loads and DER outputs has become feasible [24, 27], enabling more active participation of residential households in grid management. Users are already able to adjust their daily consumption patterns with the emergence of commercial household storage technology [4] and solar panels. This will be further facilitated by the introduction of dynamic time dependent pricing [13].

To realize an energy management framework for residential households, load scheduling has been widely studied [5, 21, 26]. The goal is to schedule controllable loads across the day to minimize the cost of procuring electricity in response to dynamic prices. In this work, we revisit the problem with the following enhancements:

- We incorporate DERs including PV and ESS. This drastically alters the energy usage pattern of the households and requires more sophisticated decision making strategies.
- We account for the net impact of individual residential scheduling decisions under a common transformer. Specifically, we ensure that at any given time period, the physical capacity constraint of the transformer is not violated due to the households collectively drawing excessive power.

Similar to previous works [22, 23], we assume unknown system dynamics, wherein we need to predict PV generation and dynamic pricing for the next time step given history, for decision making.

Reinforcement learning algorithms such as policy gradient and Q-learning have been widely used to solve load scheduling problem with unknown system dynamics [16, 17, 22]. Multi-agent reinforcement learning (MARL) is used when multiple residents with different scheduling tasks interact in the same environment with global system constraints. Independent learners (IL) and centralized method has been studied and compared [23, 26]. However, the IL approach fails to strictly prevent constraint violations while the centralized method suffers from exponential growing of states and actions. In this work, we proposed a cooperative MARL framework for real-time residential load scheduling which cooperates to minimize the cost to procuring electricity and ensure the capacity constraints of the transformers are not violated. First, we treat each resident as an agent and train independent learners using Proximal Policy Optimization (PPO) [30] with various local capacity budgets. To enforce global transformer capacity constraints, we propose an efficient cooperative algorithm that combines outcomes from different agents learned by each household. The contributions of this paper are as follows:

- We explicitly model the residential households containing PV, storage and controllable loads. We train Independent Learners (IL) for each household. We then perform real-time inferencing of control actions using the most updated state information (price, load demand, PV output etc.).
- We develop a novel low complexity knapsack based cooperation agent which combines the control actions of ILs to minimize collective cost of operation for all the households while ensuring the transformer constraints are satisfied.

Figure 1: A general reinforcement learning setting with neural network policy

- Simulation results show that our approach achieves significant cost improvement over centralized reinforcement learning and day-ahead planning baselines without capacity constraint violation.
- We measure the training and inference time by ranging the number of households from 1 to 25 and results show that our cooperative MARL approach scales best among various approaches.

2 BACKGROUND

2.1 Markov Decision Process

Markov Decision Process (MDP) [20] is a model for sequential decision making. Typically, a MDP $M = (S, A, T, \gamma, R)$, where S is the state space, A is the action space, $T : S \times A \times S \rightarrow [0, 1]$ is the transition probability model, γ is the discount factor and $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function. Policy is a set of functions $\pi(a|s) : S \rightarrow A$ and the goal of MDP is to obtain a policy such that $\sum_{t=0}^{\infty} \mathbb{E}[\gamma^t R(s_t, a_t, s_{t+1})]$ is maximized. The state transition satisfies Markov property: the next state only relies on the previous state and the current action. In real world, states are not always observable. For example, the PV generation depends on the location of the PV panel and the weather condition. These latent information constitutes the state which we don't have access to. What we observe is the PV generation value in the past, where Markov property doesn't hold. Such setting is called Partially Observable Markov Decision Process (POMDP) [29]. In this paper, we model the household environment as a POMDP.

2.2 Deep Reinforcement Learning

Reinforcement learning (RL) [32] is a learning based approach to solve MDP with unknown transition model. We show a general reinforcement learning setting with neural network to represent policy in Figure 1. Model-free RL tries to directly learn policy based on the reward signal by interacting with the environment. In this work, we apply Proximal Policy Optimization (PPO) [30] to learn independent learners. It uses clipped surrogate objective to approximate the policy gradient as

$$\nabla_{\theta} J(\theta) = \mathbb{E}[\nabla_{\theta} (\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t))] \quad (1)$$

where $r_t(\theta) = \frac{\pi_{\theta}}{\pi_{\theta_{old}}}$ and $A_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} R(s_{t'}, a_{t'}) - V(s_t)$. A_t is the advantage of action a_t over a state-dependent baseline $V(s_t)$, which

Figure 2: A multi-agent reinforcement learning setting

indicates the value of current state. In the process of training agent using PPO, we will also fit $V(s_t)$ with Monte-Carlo sampling to obtain an unbiased value function approximator. The value function is then used to perform cooperation among multiple agents, which will be illustrated in Section 5.2.

2.3 Multi-Agent Reinforcement Learning

In this work, we consider fully cooperative multi-agent interacting in the same environment [8]. It can be represented as tuple $(S_1, \dots, S_n, A_1, \dots, A_n, T, \gamma, R)$. In Figure 2, we illustrate a multi-agent reinforcement learning setting with 3 agents, where each agent interacts within its own sub environment with global action constraint denoted as $f(a_1, a_2, \dots, a_n) \leq C$ and global reward function denoted as $R(r_1, r_2, \dots, r_n)$. The advantages of this model are:

- Each sub environment can be homogeneous or heterogeneous. For example, each household can have different load types, PV panel and scheduling priorities.
- Priorities among households can be injected into the global reward function by taking a weighted sum of sub reward denoted as

$$\sum_{n=1}^N w_n \sum_{t=0}^{\infty} \mathbb{E}[y^t R(s_{t,n}, a_{t,n}, s_{t+1,n})]$$

where w_n is the weight of reward of agent n . This is practical because some households may have more important loads than others.

In our simulation, we consider simplified scenarios with homogeneous sub environments and equally weighed rewards.

2.3.1 Independent Learner (IL). In this approach, each agent independently learns its own policy by ignoring the state and action space of the other agents. While this results in the adoption of local optimal policy for each agents, however, collectively the agents might potentially violate the global environment constraints. Such scenarios require explicit cooperation among agents, where some agents need to sacrifice for the sake of collective benefit.

2.3.2 Centralized Agent. In this approach, a centralized agent learns the policy for all the agents to enable cooperation by considering states and actions from all the agents [23]. However, centralized agents suffer from exponential increase of state and action space as the number of agent increases [18].

As both the methods discussed above have their own limitations, we develop a novel approach by training Independent Learners for each household (Section 5.2.1) and using a low computational

Figure 3: Smart grid network model

complexity cooperation algorithm to ensure satisfaction of global environment constraints (Section 5.2.2).

3 RELATED WORK

Smart energy management has been widely studied with the proliferation of DERs and IoT technology. In [21], the authors model electricity market and residents as two players. The residential scheduling problem is modeled as a Markov Decision Process (MDP) and game theoretic techniques are used for solution. However, their model is only applicable to small number of residents due to limited scalability. In [26], the authors assume Markovian price and residential device usage with known transitions and use Q-learning to learn optimal strategy to shift the loads with respect to dynamic pricing. In [5], the authors model appliance operation scheduling problem as Mixed-Integer Linear Programming (MILP) and proposes a heuristic to efficiently approximate the optimal solution. In [22], the authors utilize Deep Q-Network (DQN) and Deep Policy Gradient (DPG) to perform online energy scheduling to minimize cost and reduce peak loads. In [25], the authors use MILP and a dynamic scheduling model for optimal load scheduling among different consumer types for different load types to find optimal storage requirement for a microgrid under different scenarios. In their problem, the storage requirement is passively computed rather than actively participating in the optimization problem.

The innovation in our problem formulation is that actively utilize storage and schedule loads under dynamic pricing and stochastic PV generation. Furthermore, we also consider the cooperation among individual households with different load profile connected to the same transformer with a capacity constraint. Simply solving this problem using MILP will not be scalable due to exponential increase in the size of state and action space. Hence, we consider training multiple independent learners for each household with various local constraints offline using historical data. Then, we use the trained model to perform online inferencing. To enable cooperation to satisfy global constraints, we propose an efficient online algorithm to select local actions for each household that achieves global optimum.

4 PROBLEM SETUP

4.1 Overview

In this work, we adapt the smart grid model from [19] as illustrated in Figure 3. Power from the high voltage transmission network is fed into the low voltage network using HV/LV sub-station. Several Low

Figure 4: Single household model. The arrow denotes the power flow direction.

Voltage (LV) feeders originate from the sub-station to deliver power. Each LV feeder is connected to several transformers using laterals. The transformers further step down the voltage and provide power to households using distributors. The multi-agent framework in this work focuses on the group of residents under the same transformer with each one of them treated as a single agent.

4.2 Single Household Energy Model

Figure 4 illustrates our household model – PV panel, storage system with bidirectional energy flow and a list of controllable loads.

4.2.1 Energy Storage Model. We consider batteries as the main ESS in household [4]. Let $P_{b,n}$ denote charging and discharging power for household n . We assume the battery can work in charging, discharging or disconnected mode. Let $E_{b,n}(t)$ denotes the energy stored in battery at time t . The battery energy transition is given as

$$E_{b,n}(t+1) = \eta_{b,n}E_{b,n}(t) + \lambda_{b,n}(t)P_{b,n}\Delta t \quad (2)$$

where $\eta_{b,n} \in (0, 1]$ denotes battery efficiency and $\lambda_{b,n}(t) = 1/-1/0$ denotes charging/discharging/disconnected, respectively.

4.2.2 Household Loads. We follow the same household loads categorization as in [25]:

Modifiable power loads. Modifiable power loads need to be served in a fixed window where the total energy requirement is met. They are characterized by earliest starting time, a deadline and the required number of operations during this period. Examples are EV charging and HVAC operation. In this paper, we focus on scheduling modifiable power loads with the presence of PV and storage. For household n , the scheduling task is M interruptible loads with operation power $P_{n,1}, P_{n,2}, \dots, P_{n,M}$ and required number of operation times $L_{n,1}, L_{n,2}, \dots, L_{n,M}$, where $L_{n,m} < 96, m = 1, 2, \dots, M$. For simplicity, we assume the earliest starting time is the beginning of the day with an end of day deadline. At each timestamp, the action space for household n load scheduling is a binary vector

$$a_n(t) = (a_{n,1}(t), a_{n,2}(t), \dots, a_{n,M}(t)) \quad (3)$$

where $a_{n,m} = 1/0$ denotes turning on/off load m , respectively.

Time shiftable loads. Time shiftable loads have fixed operation power and length and once started, should not be interrupted (for example, dish washers and laundry units). Accurate prediction of dynamic price is the key to scheduling time shiftable loads. Time

shiftable loads can be viewed as modifiable power loads with additional constraints: they should be scheduled in consecutive time slots until finished. Thus, they are easily included in our framework.

Non-shiftable loads. Non-shiftable loads (eg. microwaves, desktops) are requested on demand. It can be viewed as modifiable power loads with only one action - on. Thus, it can also be easily included in our framework.

4.2.3 PV generation and Dynamic Pricing. The PV generation at each time step is determined by the area of solar panels, the weather condition, the time of the day and so on. The dynamic pricing in the electricity market is determined by the balance between buyers and sellers. We assume that the household can only observe the history value of PV generation and dynamic pricing without knowing the underlying system dynamics. For day ahead scheduling, we are required to predict the PV generation and the dynamic pricing for the next day. For online scheduling, we can incorporate online information with historical data to make more accurate prediction for decision making.

4.3 Transformer Capacity Constraint

In this section, we formally define the physical grid constraints that our cooperative algorithms enforces. Let K be the set of transformers connected to the LV feeder using laterals. For each transformer $k \in K$, let $H(k)$ be the households connected to the transformer using distributors. Let $P_{grid,h}(t)$ be the power demand of household $h \in H(k)$ at time t . Henceforth, for simplicity, we will use $E_{grid,h}(t) = P_{grid,h}(t) \times \Delta t$, where Δt is the interval size i.e. the energy demand in our model. Hence, for each transformer $k \in K$, the following constraint needs to be satisfied:

$$|\sum_{h \in H(k)} E_{grid,h}(t)| \leq C_k \quad (4)$$

where C_k is the capacity rating of the transformer k . From Section 4.4 onwards, we will focus on a single transformer and will drop k to simplify notation.

In addition to the transformer capacity constraints, the utility also needs to ensure that voltages are within the threshold. A high load demand on LV feeders can cause low voltages downstream i.e. at the transformers further away from the HV/LV substation. By solving power flow equations [12], utility can dynamically determine real and reactive power (energy) limits at each transformer to limit voltage fluctuations which can be modeled as follows:

$$p^{min}(t) \leq \sum_{h \in H(k)} P_{grid,h}(t) \leq p^{max}(t) \quad (5)$$

$$Q^{min}(t) \leq \sum_{h \in H(k)} Q_{grid,h}(t) \leq Q^{max}(t) \quad (6)$$

where $E_{grid,h}(t) = P_{grid,h}(t) + iQ_{grid,h}(t)$ and $p^{min}/max(t) + iQ^{min}/max(t)$ is the time dependent complex energy limits. However, since utilities just actively deliver real power with households being charged only for real power and reactive power injection for voltage regulation is handled by the utility [6], we only focus on the real power constraint in this work. Our framework will enforce

either transformer capacity constraint or voltage fluctuation limiting constraints, whichever is smaller. To simplify discussion, we will simply denote the constraint as $C(t)$.

4.4 Markov Decision Process Setup

We formulate our problem as a multi-agent Markov Decision Process (MDP) with N agents (households). We assume the optimization horizon is one day with T timestamps. In our simulation, we set $T = 96$ (15 minutes interval) as in most previous work [5, 22].

4.4.1 State Space. Each household has its own **local** state and shares **global** states. We further divide states into **fully observable** states and **partially observable states** where fully observable states satisfy Markovian property while partially observable states don't. For household index n ,

- **Local fully observable states:** At time t , the remaining load vector $(l_{n,1}(t), l_{n,2}(t), \dots, l_{n,M}(t))$ and energy storage $E_{b,n}(t)$.
- **Local partial observable states:** Historical PV generation before time step t : $pv_n(0), pv_n(1), \dots, pv_n(t-1)$.
- **Global partial observable states:** Historical unit price of energy before time step t : $c(0), c(1), \dots, c(t-1)$.

Initial state. The remaining load $l_{n,m}(1) = L_{n,m}, \forall n, m$. We assume random initial storage $E_{b,n}(1), \forall n$. During simulation, we use real data to generate household PV generation and unit energy price.

4.4.2 Action Space. For household index n , the action space is

- The binary vector $a_n(t)$ to turn on/off each load as defined in Equation 3.
- The battery operation mode $\lambda_{b,n}(t)$ as defined in Equation 2.

4.4.3 Transition Model. First, the energy demand and supply should match

$$\sum_{i=1}^M a_{n,m}(t)P_{n,m} + \lambda_{b,n}(t)P_{b,n} = pv_n(t) + \frac{E_{grid,n}(t)}{\Delta t}, \forall n, t \quad (7)$$

where $E_{grid,n}(t)$ is the energy drained from the smart grid. We assume $E_{grid,n}(t)$ can be negative, in which case we are selling energy to the market using storage. The transition model of the remaining load for household n load m is

$$l_{n,m}(t+1) = l_{n,m}(t) - a_{n,m}(t), \forall m, n \quad (8)$$

The transition model of storage is defined in Equation 2. The transition model of PV and price is determined by the underlying system dynamics and we only observe their values. In our simulation, we use real world historical values directly without explicitly modeling the data generation process.

4.4.4 System Constraints.

Local. For each household, the total number of time slots turning on certain loads is equal to the requirement.

$$\sum_{t=1}^T a_{n,m}(t) = L_{n,m}, \forall n, m \quad (9)$$

Algorithm 1: Offline Optimization (Day-ahead Planning)

Predict: $pv_n(t), c(t), \forall n = 1, \dots, N, t = 1, \dots, T$;
Solver: Run ILP to solve minimization problem defined by Equation 11, subject to Equation 2-10;
Result: $a_{n,m}(t), \lambda_{b,n}(t), \forall n = 1, \dots, N, m = 1, \dots, M, t = 1, \dots, T$
Online Execution:
for $t = 1$ **to** T **do**
 Take planned action
 $a_{n,m}(t), \lambda_{b,n}(t), \forall n = 1, \dots, N, m = 1, \dots, M$
end

Global. At any time, the total energy drained from the households connected to a transformer must not exceed the transformer capacity:

$$|\sum_{n=1}^N E_{grid,n}(t)| \leq C(t), \forall t \quad (10)$$

For simplicity, we assume no lower bound in the drawn energy.

4.4.5 Objective. We want to minimize the cost of draining energy from the grid to finish the load scheduling as well as satisfying the feeder capacity. The goal is to choose $a_n(t)$ and $\lambda_{b,n}(t)$ for $n = 1, 2, \dots, N$ and $t = 1, 2, \dots, T$ such that the total cost J is minimized, where J is calculated as

$$J = \sum_{t=1}^T \sum_{n=1}^N E_{grid,n}(t)c(t) \quad (11)$$

subject to Equation 2-10.

5 APPROACH

Before we delve into our proposed cooperative RL based solution in Section 5.2, we briefly discuss three baseline algorithms: i) offline optimization, ii) online optimization, iii) centralized reinforcement learning agent

5.1 Baseline Approaches

5.1.1 Offline Optimization. Offline optimization is also known as **day-ahead planning**. We show the algorithm outline in Algorithm 1. First, we predict all day-ahead uncertainties including PV generation and dynamic energy pricing. Then, we optimize the total cost J subject to constraints 2-10 using Integer Linear Programming (ILP). The main advantage of this approach is that it supports real-time load scheduling very well since no real-time computation is needed. However, this approach has three main drawbacks:

- The day-ahead predictor can predict the trend accurately but suffer from real-time information blindness. For example, the PV generation at 1pm has not only strong correlation with PV generation at 1pm during previous days, but also 12pm generation of the same day.
- It is open-loop control with no integration of real-time information. Thus, the performance is greatly impacted if irregular scenarios happen.
- ILP is NP-complete in general and doesn't scale very well w.r.t the number of households.

Algorithm 2: Online Optimization (Online Planning)

Input: Historical PV and dynamic energy pricing data;
Train predictor $\bar{p}v_n(t)$, $\forall n$ and $\bar{c}(t)$ offline;
Online Execution:
for $t = 1$ **to** T **do**
 Predict $p v_n(k)$, $c(k)$, $\forall n = 1, \dots, N$, $k = t + 1, \dots, T$;
 Run ILP and obtain next time step action
 $a_{n,m}(t + 1)$, $\lambda_{b,n}(t)$, $\forall n = 1, \dots, N$, $m = 1, \dots, M$;
 Take planned action
 $a_{n,m}(t + 1)$, $\lambda_{b,n}(t)$, $\forall n = 1, \dots, N$, $m = 1, \dots, M$;
 Observe ground truth PV generation and price at $t + 1$;
end

5.1.2 Online Optimization (Online Planning). We show a skeleton of the online optimization in Algorithm 2. The online optimization will also generate a sequence of planned action for the entire day as offline optimization. **However, only the next time step action will be executed.** Then, the ground truth PV generation and dynamic price will be observed. The predictor will utilize newly observed ground truth to make finer prediction and the agent will re-plan. In contrast to offline approach, online planning integrates real-time information for decision making and achieves near-optimal performance as shown by simulation results. However, we have to run ILP for every time step, which loses real-time load scheduling capability.

5.1.3 Centralized Reinforcement Learning. For comparison, we also train a centralized RL agent to solve the residential load scheduling problem. To train the centralized agent, a global environment that concatenates Markov Decision Process components: observations, action, reward, transition model is implemented. In our problem, the number of action variables in a single household is K and the size of action space of N household is 2^{NK} . The large observation and action space prevents the policy from converging and yields worse performance than cooperative independent learners as shown in simulation results in Section 7.3.

5.2 Cooperative Reinforcement Learning

In this paper, we propose a multi-agent cooperative reinforcement learning algorithm to solve the optimization problem defined in Section 4. First, we discretize the capacity C into $\{0, 1, \dots, C'\}$. This can always be done if we only care about numerical solution up to finite decimals, say two. Then, the total number of possible $C' = \lceil 100C \rceil$. For each $c \in \{0, 1, \dots, C'\}$ and household n , we train Independent Learner $\mathcal{A}_{n,c}$ with local capacity c using Proximal Policy Optimization (PPO). During deployment, each agent $\mathcal{A}_{n,c}$ produce an action $a_{n,c}$ using policy network and an expected sum of future reward $r_{n,c}$ using value network. Then the objective of the cooperation algorithm is to select exactly one action from each household such that the sum of capacities is $\leq C'$ and the sum of reward is maximized. The solution to this problem is provided by the following dynamic programming formulation:

$$\Phi(c, n + 1) = \max_{c' \in \{0, 1, \dots, C'\}} \{\Phi(c - a_{n+1,c'}, n) + r_{n+1,c'}\} \quad (12)$$

Capacity	Household 1	Household 2	...	Household N
C	$a_{1,C} = \text{Policy}(s_1)$ $r_{1,C} = \text{Value}(s_1)$	$a_{2,C} = \text{Policy}(s_2)$ $r_{2,C} = \text{Value}(s_2)$	...	$a_{N,C} = \text{Policy}(s_N)$ $r_{N,C} = \text{Value}(s_N)$
$C - 1$	$a_{1,C-1} = \text{Policy}(s_1)$ $r_{1,C-1} = \text{Value}(s_1)$	$a_{2,C-1} = \text{Policy}(s_2)$ $r_{2,C-1} = \text{Value}(s_2)$	...	$a_{N,C-1} = \text{Policy}(s_N)$ $r_{N,C-1} = \text{Value}(s_N)$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
1	$a_{1,1} = \text{Policy}(s_1)$ $r_{1,1} = \text{Value}(s_1)$	$a_{2,1} = \text{Policy}(s_2)$ $r_{2,1} = \text{Value}(s_2)$	...	$a_{N,1} = \text{Policy}(s_N)$ $r_{N,1} = \text{Value}(s_N)$
0	$a_{1,0} = \text{Policy}(s_1)$ $r_{1,0} = \text{Value}(s_1)$	$a_{2,0} = \text{Policy}(s_2)$ $r_{2,0} = \text{Value}(s_2)$	...	$a_{N,0} = \text{Policy}(s_N)$ $r_{N,0} = \text{Value}(s_N)$

Figure 5: An illustration of cooperation process. We train one agent with local capacity $c \in \{0, 1, \dots, C'\}$ for each household. During deployment, we choose one agent from each household such that the sum of the reward is maximized and actions satisfy the global capacity constraint. For example, the agent with red circle for each household is selected.

Algorithm 3: Cooperative Reinforcement Learning
Offline Training:

$\text{local_capacity} \leftarrow [0, 1, (1 + \epsilon)^{\frac{1}{N}}, \dots, (1 + \epsilon)^{\frac{k}{N}}]$;

for $n = 1$ **to** N **do**

 for cap in local_capacity **do**

$\text{Policy}_{n,\text{cap}}, \text{Qvalue}_{n,\text{cap}} = \text{ppo}(\text{env}_n, \text{cap})$;

end
end

Let $\text{cap}_i \leftarrow (1 + \epsilon)^{\frac{i}{N}}$, $i = 0, 1, \dots, k$ and $\text{cap}_{-1} = 0$;

Let $E_{\text{grid},n}(t) = f(a_{n,m}, \lambda_{b,n})$;

Online Deployment:
for $t = 1$ **to** T **do**

$a_{n,m}(t), \lambda_{b,n}(t) \leftarrow \text{Policy}_{n,\text{cap}_k}(s_n(t)), m = 1, 2, \dots, M$;

if $\sum_{n=1}^N f(a_{n,m}, \lambda_{b,n}) > C$ **then**

$a_{n,m}(t), \lambda_{b,n}(t) \leftarrow$
 $\text{cooperate}(\text{Policy}_{n,\text{cap}}, \text{Qvalue}_{n,\text{cap}})$;

end

Take action $a_{n,m}(t), \lambda_{b,n}(t)$;

end

where $c \in \{0, 1, \dots, C'\}$ and $n \in \{2, \dots, N\}$ and $\Phi(c, 1)$ entries are initialized based on the available actions for the first household. Equation 12 can be solved by creating a table Φ of size $(C' + 1) \times N$ and iteratively filling entries starting from $\{0, 1\}$ till $\{C', N\}$. Filling each entry requires $O(C')$ time. Hence, the total runtime is $O(C'^2 N)$. However, the runtime and the number of agents to train offline, which is $C'N$, can be very large since $C' = \lceil 100C \rceil$. In the future subsection, we provide an approximation algorithm that reduce the number of agents to train for each household to $\log_{1+\epsilon} C'$. We show an illustration of the process in Figure 5.

5.2.1 Learning Independent Learners. In this work, we train Independent Learner (IL) using PPO algorithm [30]. The target policy network is a function map from observation to action. We use neural network with parameter θ and represent it as $\pi_\theta(a_t | s_t)$. To model the relationship between different output actions, we choose

Multivariate Gaussian Controller with mean and covariance matrix. The output is sampled from a trained Multivariate Gaussian distribution. We discretize the output by treating positive value as 1 and negative value as 0.

To train independent learners, we use large negative reward signal to guide the agent to avoid violating local capacity constraint. When training independent learners, the reward signal and observation is too noisy for the agent to converge to a good solution. To deal with this problem, we adopt the idea from AlphaGo [31]. Instead of directly training ILs, we obtain around 1000 optimal actions for each household by running ILP using ground truth PV generation and price information. We train our policy network using optimal action in a supervise way (In AlphaGo, the supervision comes from human players). Then, we finetune the policy network with reward signals using reinforcement learning algorithm Proximal Policy Optimization [30].

5.2.2 Approximation Algorithm for Cooperation with Bounded Capacity Constraint Violation. One technique to reduce the time complexity of cooperation algorithm (12) is to develop approximation algorithm similar to one developed in [11]. We choose a parameter ϵ and define the following partitions for the range $[0, \dots, C']$:

$$(0, 1], (1, (1+\epsilon)^{\frac{1}{N}}], ((1+\epsilon)^{\frac{1}{N}}, (1+\epsilon)^{\frac{2}{N}}], \dots, ((1+\epsilon)^{\frac{k-1}{N}}, (1+\epsilon)^{\frac{k}{N}}]$$

where $(1+\epsilon)^{\frac{k-1}{N}} < C' \leq (1+\epsilon)^{\frac{k}{N}}$ and hence, $k = \lceil N \log_{1+\epsilon} C' \rceil$. The partitions are indexed using $k_p \in \{0, 1, \dots, k\}$. Let $\hat{c}(k_p)$ be the upper value of the partition indexed using k_p . Now, we calculate $r_{n,c}, ac_{n,c} \forall n \in [N]$ and $c \in \{\hat{c}(0), \hat{c}(1), \dots, \hat{c}(k)\}$. The solution to this problem is now provided by the following dynamic programming formulation:

$$\tilde{\Phi}(\tilde{c}, n+1) = \max_{k_p \in \{0, 1, \dots, k\}} \{ \tilde{\Phi}(\tilde{c} - ac_{n+1, c(k_p)}, n) + r_{n+1, c(k_p)} \} \quad (13)$$

where $\tilde{c} \in \{\hat{c}(0), \hat{c}(1), \dots, \hat{c}(k_p)\}$ and $n \in \{2, \dots, N\}$. Note that the expression $\tilde{c} - ac_{n+1, c(k_p)}$ is mentioned for simplicity of notation. In implementation, given $\tilde{c} + c(k_p)$, the entry $\tilde{c}'$ such that $\tilde{c}' \geq \tilde{c} + c(k_p) > \frac{\tilde{c}'}{(1+\epsilon)^{\frac{1}{N}}}$ is updated. The size of the table for Equation 13 is $(k+1) \times N$ and each entry requires $O(k)$ time. Hence, the time complexity is $O(N^3 \log_{1+\epsilon}^2 C')$ which is polynomial in the encoded input size. Moreover, we have the following theorem for Equation 13.

THEOREM 5.1. *Equation 13 is a polynomial runtime complexity algorithm to perform cooperation whose solution maximizes reward while in the worst case violates the capacity constraint C' by $(1+\epsilon)$.*

PROOF. Similar to the analysis in [11] we will prove the theorem using induction. For each entry c, n , we will show that there exists $\tilde{c}, n$ such that the following holds:

- (1) $\tilde{\Phi}(\tilde{c}, n) \geq \Phi(c, n)$
- (2) $\tilde{c} \leq (1+\epsilon)^{\frac{n}{N}} c$

For $n = 1$, it is trivially true as c will be rounded up to a value $\tilde{c} \leq (1+\epsilon)^{\frac{1}{N}} c$ and $\tilde{\Phi}(\tilde{c}, 1)$ will choose the maximum from all c mapped to $\tilde{c}$. Assuming the induction holds for n , for $n+1$, let c_{n+1} be the capacity of the action chosen from household $n+1$. Let $\frac{\tilde{c}'}{(1+\epsilon)^{\frac{1}{N}}} < \tilde{c} + c_{n+1} \leq \tilde{c}'$ i.e. $\Phi(\tilde{c}', n+1)$ is the updated entry

due to this action. By induction condition (2) and adding c_{n+1} we have, $\frac{\tilde{c}'}{(1+\epsilon)^{\frac{1}{N}}} + c_{n+1} \leq c + c_{n+1}$. So, $\tilde{c}' < (1+\epsilon)^{\frac{1}{N}} (\tilde{c} + c_{n+1}) \leq (1+\epsilon)^{\frac{1}{N}} (\tilde{c}' + (1+\epsilon)^{\frac{n}{N}} c_{n+1}) \leq (1+\epsilon)^{\frac{n+1}{N}} (\tilde{c} + c_{n+1})$ as $(1+\epsilon)^{\frac{n}{N}} > 1$. So, $\tilde{c}' \leq (1+\epsilon)^{\frac{n+1}{N}} (\tilde{c} + c_{n+1})$ and thus the induction condition (2) holds. Induction condition (1) holds as $\tilde{\Phi}(\tilde{c}', n+1)$ will pick the maximum from all $c + c_{n+1}$ mapped to it.

Now, substituting $n = N$ in induction (2), we get the theorem. $\square$

Remark 1: Note that for Equations 12 and 13, the values of c are assumed to be integers. This is without loss of generality as fractional values can be scaled to represent them using integers. Moreover, note that the optimality of Equation 13 is with respect to the values output by the independent learners used as input to Equation 12. The output of the independent learners are treated as ground truth and its sub-optimality is not considered in the algorithm.

5.2.3 Heuristic for Cooperation with Strict Capacity Constraint Satisfaction. The runtime of the approximation algorithm developed in Section 5.2.2 depends upon ϵ i.e. the factor by which the capacity constraint is violated. For example, choosing $\epsilon = 1$ for a runtime of $O(N^3 \log_2 C')$ will lead to a capacity constraint violation of $2C'$ in the worst case. For scenarios which require low computational complexity but cannot tolerate such high errors, we develop a heuristic that ensures that the constraints are never violated. The heuristic however, does not guarantee optimal solution.

For each n , using $\lceil \log_2 C' \rceil$ independent learners, the heuristic computes $r_{n,c}, ac_{n,c}$ for $c \in \{C', \frac{C'}{2}, \frac{C'}{2^2}, \dots, \frac{C'}{2^k}\}$, where $\frac{C'}{2^k} \geq 1 > \frac{C'}{2^{k+1}}$. We now have the following dynamic programming formulation for cooperation:

$$\Theta(c, n+1) = \max_{c' \in \{C', \frac{C'}{2}, \frac{C'}{2^2}, \dots, \frac{C'}{2^k}\}} \{ \Theta(c - ac_{n+1, c'}, n) + r_{n+1, c'} \} \quad (14)$$

The size of the table Θ is $O(C' \times N)$ and each entry requires $O(\log_2 C')$ runtime. Hence, the computational complexity of the heuristic is $O(C'N \log_2 C')$.

Remark 2: Note that the dynamic programming formulations above cannot handle negative values. To handle this, if some households are injecting surplus power into the distributors and others are drawing power, and wlog, power drawn is more than the power injected, then we will solve the above problem only for the load consuming households with increased capacity $C' + P_s$, where P_s is the power injected by the surplus power households.

6 EXPERIMENTAL SETUP

6.1 Dataset

6.1.1 PV Generation and Dynamic Pricing. We use mixed real and synthetic datasets to conduct experiments. The Pecan Street dataset [2] records the energy consumption and PV generation for users from 2013-01-01 to 2016-12-31 every 15 minutes. We choose a subset of 25 users with PV panels. PJM dataset [3] records the dynamic pricing in the east coast every hour. We interpolate the data using cubic interpolation to generate 15-minute dynamic pricing.

Figure 6: Online/offline Long Short-Term Memory (LSTM) predictor for PV generation and dynamic pricing

6.1.2 Load Profile. We pick 5 typical household appliances from [9] and EVs from [10] and summarize the attributes in Table 1. We generate load profiles for 25 households by uniform sampling from the minimum value and the maximum value.

6.1.3 Storage Profile. The household storage capacity ranges from 9-12 kWh. We generate 25 household storage capacity profiles by uniform sampling from [9, 12]. The capacity efficiency η is set to 0.99 and the battery input/output power is set to half of the storage capacity.

6.1.4 Dynamic Pricing Normalization. In this problem, the load scheduling horizon is one-day. Thus, only the relative price within a day matters. In simulation, we preprocess the price to have maximum of 1 and minimum of 0.

6.1.5 Transformer Capacity. In training independent learners, we find that very few agents violate local capacity 4 kWh and most of them will violate local capacity 2 kWh. Thus, we consider two sets of transform capacity that leads to different environment types.

In **loose** environment setting, we set transform capacity to be $4N$, where N is the number of households. In this type, very few cooperation is needed and each agent can act greedily.

In **compact** environment setting, we set transformer capacity to be $2N$, where N is the number of households. We expect explicit cooperation among agents to minimize the number of violations and maximize the reward at the same time. When training Independent Learners, we choose $C' = C$ and $c' = 2, 4, 8, 15, \infty$.

6.2 Predictor Architecture

An accurate predictor is key to making good decisions. In this work, we use Long Short-Term Memory (LSTM) [15] as the predictor architecture (Figure 6). The online and offline predictors share the same architecture which takes in historical data as input, passed into LSTM cell and generates the next time step output. An LSTM cell contains input gate and forgot gate. For offline predictor, x_{t-i} represents the data i^{th} day ahead. It is a 96 dimension vector. For online predictor, x_{t-i} represents $i - th$ time step ahead and it is a scalar. Data from 2013-01-01 to 2015-12-31 was used to train the predictor with a train/val split ratio of 0.75/0.25.

6.3 Baseline Models

To evaluate the performance of our cooperative RL, we set up three baseline methods for comparison.

- (1) We train offline predictor for PV generation for each household and dynamic pricing using LSTM. We use the predicted

Figure 7: PV prediction from various predictor

Figure 8: Dynamic pricing prediction from various predictor

generation and prices to perform day-ahead planning by running Integer Linear Programming (ILP) using Gurobi Optimization Software [14].

- (2) We train online predictor for PV generation for each household and dynamic pricing using LSTM. Then, we perform online planning i.e. planning for the next interval by running Integer Linear Programming (ILP) using Gurobi Optimization Software [14].
- (3) We obtain optimal action by feeding ground truth PV generation and dynamic pricing to ILP optimizer using Gurobi Optimization Software [14].

6.4 Simulation Environment

We build a simulation environment based on OpenAI Gym [7] to evaluate and compare the performance of various approaches. The agents are trained using PyTorch framework [28]. We use data from 2013-01-01 to 2015-12-31 to train agents and use data from 2016-01-01 to 2016-12-31 to test.

Load Name	Min Power (kW)	Max Power (kW)	Min Slots (15 min)	Max Slots (15 min)
Dishwasher	1.2	1.5	2	8
Washing Machine	0.5	1	4	8
Clothes Dryer	1	4	4	8
Home Air Conditioner	1	4	48	80
Electric Vehicle	2.5	7	32	48
Vacuum Cleaner	0.2	0.7	4	8

Table 1: Load Profiles for Experiments

7 RESULTS

7.1 Offline Predictor vs. Online Predictor

We show the prediction results for PV generation of a single household and dynamic pricing during a single day in Figure 7 and Figure 8. Results show that online prediction has much lower error versus day-ahead predictor. This indicates more accurate planning by online approach. However, the online optimization needs to re-plan by running ILP at each time step, which is computational expensive. By training RL agents offline, our approach takes advantage of the accurate prediction from online predictor, while avoiding high computation as inferencing has low computational requirement and cooperation algorithm is very efficient.

7.2 Independent Learner

7.2.1 Insights of Learned Strategy. We show an example of the learned strategy by running offline optimization, online optimization and reinforcement learning agent to perform load scheduling and record status including storage, absolute grid consumption and dynamic pricing. The status curves are shown in Figures 9, 10 and 11, respectively. A reasonable strategy is to charge the storage at low price period and discharge to supply the loads when price is high. As shown in Figure 10 and Figure 11, the agent trained using reinforcement learning resembles the action of online optimization with less noisy behavior and successfully catches the peak time of the price. In Figure 9, the peak of offline predicted price mismatches with the ground truth peak, yielding storage charging at high price and thus increasing the overall cost. The costs of offline optimization, online optimization and reinforcement learning independent learner in this example are 14.27, 4.95 and 5.74, respectively.

7.2.2 Local Constraint. To show the effectiveness of the reward signal based local constraint for independent learner, we show the results of absolute grid energy value with various local capacities in Figure 12. According to the figure, all the independent learners obey the local capacity constraint. It is also worth noticing that increasing the local capacity from 8 to 15 doesn't change the grid energy curve very much, which indicates the consistency of the optimal policy learned by RL.

7.3 Performance Comparison

We compare the performance of the proposed cooperative RL vs. several baselines mentioned in Section 6.3 by using these methods to perform load scheduling for two consecutive weeks. The normalized average household cost of various approaches are shown in Figure 13 and Figure 14. In both environments, the cooperative RL

Figure 9: Independent learner storage, absolute grid energy vs. dynamic pricing obtained by offline optimization (day-ahead planning)

Figure 10: Independent learner storage, absolute grid energy vs. dynamic pricing obtained by online optimization (online planning)

approach performs much better than centralized RL and slightly better than offline optimization. The online optimization approach almost performs optimally due to low prediction error.

The cost of all the algorithms increases when the transformer capacity decreases as shown in Figure 14. The cost of the non-cooperative agent is less than the cooperative agent. This is expected since non-cooperation acts greedily but significantly increases the number of violations as shown in Figure 15. Generally, reinforcement learning based agents have fewer violations than

Figure 11: Independent learner storage, absolute grid energy vs. dynamic pricing trained by reinforcement learning algorithm (proximal policy optimization)

Figure 12: Absolute grid energy value with various local capacity trained with reinforcement learning agent

Figure 13: Performance of various algorithms of two weeks execution in loose transformer capacity environment

Figure 14: Performance of various algorithms of two weeks execution in compact transformer capacity environment

Figure 15: Average daily number of violations of various algorithms of two weeks execution in compact transformer capacity environment

offline optimization. We didn't include online optimization and optimal in Figure 15 since the number of violation is zero.

8 DISCUSSION

8.1 Scalability

As shown in figures 13-15, the average household cost and number of violations remains almost the same for the cooperative RL approach. For offline optimization, the number of violations increases with the number of households due to the worsening impact of peak mismatch. For online optimization, the computation time required by Integer Programs often lasts from 20 minutes to hours, thus reducing their real-time utility. Finally, the exponential growth in the observation and action space of the centralized approach prevents the algorithm from converging. Hence, distributed RL/optimization approaches would appear more suitable for efficient cooperative algorithms. Following this design logic, we train distributed learners using RL and also propose an efficient algorithm that combines the results while satisfying global constraints. This achieves a good tradeoff between offline training complexity and online execution

speed. More importantly, all these independent learners can be trained in parallel, which further yields better scalability.

8.2 The Applicability of RL Approach

Compared with optimization approaches i.e. approaches which perform a prediction and then solve the optimization problem, machine learning (ML) based approaches are much faster at run time. The reason is that ML based approaches are trained offline using historical data and only inferencing is performed at runtime using the most updated environment information. Inferencing from a trained model requires low computational complexity (as compared to solving an optimization problem). However, the performance of ML based algorithms relies on the generalization from the training data to the testing data. In this work, we ensure generalization by sampling the load profiles for both testing and training from the same distribution. However, this may not be true in real scenarios, where the load patterns of a single user may change over time. This requires effective detecting mechanism and retraining of the policy based on new patterns.

9 CONCLUSION AND FUTURE WORK

In this work, we addressed the problem of cooperative decision making from a collection of agents. We limit computational complexity while ensuring cooperation by training Independent Learners (ILs) and combining their results using a knapsack based agent.

The current work considers a reward function dependent only on the load consumption and price of electricity. In our future works, we will use more sophisticated reward functions which also consider customer satisfaction. This can be formulated using hierarchical MDP. Moreover, we will also consider fairness of cooperation—the losses incurred over a period of time by agents forced to adopt sub-optimal actions should be equitably distributed among the households. We will develop online cooperation algorithm with bounded competitive ratios.

REFERENCES

- [1] 2012. SunShot Vision Study: A Comprehensive Analysis of the Potential for U.S. Solar Electricity Generation. <https://digital.library.unt.edu/ark:/67531/metadc836555/>.
- [2] 2018. Pecan Street Dataset. <http://www.pecanstreet.org/category/dataport/>.
- [3] 2018. PJM Dataset. <http://www.pjm.com/markets-and-operations/>.
- [4] 2018. Tesla Powerwall: the complete battery review. <https://www.energysage.com/solar/solar-energy-storage/tesla-powerwall-home-battery/>.
- [5] A. Agnetis, G. de Pascale, P. Detti, and A. Vicino. 2013. Load Scheduling for Household Energy Consumption Optimization. *IEEE Transactions on Smart Grid* 4, 4 (Dec 2013), 2364–2373. <https://doi.org/10.1109/TSG.2013.2254506>
- [6] Hamed Ahmadi, José R Martí, and Hermann W Dommel. 2015. A framework for volt-VAR optimization in distribution systems. *IEEE Transactions on Smart Grid* 6, 3 (2015), 1473–1483.
- [7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. [arXiv:arXiv:1606.01540](https://arxiv.org/abs/1606.01540)
- [8] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. *Multi-agent Reinforcement Learning: An Overview*. Springer Berlin Heidelberg, Berlin, Heidelberg, 183–221. https://doi.org/10.1007/978-3-642-14435-6_7
- [9] DaftLogic. 2018. List of the Power Consumption of Typical Household Appliances. <https://www.daftlogic.com/information-appliance-power-consumption.htm>.
- [10] Ergon Energy. 2018. Charging your electric vehicle. <https://www.ergon.com.au/network/smarter-energy/electric-vehicles/charging-your-electric-vehicle>.
- [11] Thomas Erlebach, Hans Kellerer, and Ulrich Pferschy. 2002. Approximating multiobjective knapsack problems. *Management Science* 48, 12 (2002), 1603–1612.
- [12] Masoud Farivar and Steven H Low. 2012. Branch flow model: Relaxations and convexification. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 3672–3679.
- [13] Ahmad Faruqui and Sanem Sergici. 2010. Household response to dynamic pricing of electricity: a survey of 15 experiments. *Journal of regulatory Economics* 38, 2 (2010), 193–225.
- [14] LLC Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. <http://www.gurobi.com>
- [15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- [16] B. Kim, Y. Zhang, M. van der Schaar, and J. Lee. 2016. Dynamic Pricing and Energy Consumption Scheduling With Reinforcement Learning. *IEEE Transactions on Smart Grid* 7, 5 (Sept 2016), 2187–2198. <https://doi.org/10.1109/TSG.2015.2495145>
- [17] Sunyong Kim and Hyuk Lim. 2018. Reinforcement Learning Based Energy Management Algorithm for Smart Energy Buildings. *Energies* 11, 8 (Aug 2018), 2010. <https://doi.org/10.3390/en11082010>
- [18] Jelle R. Kok and Nikos Vlassis. 2004. Sparse Cooperative Q-learning. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML '04)*. ACM, New York, NY, USA, 61–. <https://doi.org/10.1145/1015330.1015410>
- [19] Stephen Lee, Srinivasan Iyengar, David Irwin, and Prashant Shenoy. 2017. Distributed Rate Control for Smart Solar Arrays. In *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy '17)*. ACM, New York, NY, USA, 34–44. <https://doi.org/10.1145/3077839.3077840>
- [20] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming (1st ed.)*. Wiley.
- [21] S. Misra, A. Mondal, S. Banik, M. Khatua, S. Bera, and M. S. Obaidat. 2013. Residential Energy Management in Smart Grid: A Markov Decision Process-Based Approach. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. 1152–1157. <https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.200>
- [22] E. Mocanu, D.C. Mocanu, H.P. Nguyen, A. Liotta, M.E. Webber, M. Gibescu, and J.G. Slootweg. 2018. On-line building energy optimization using deep reinforcement learning. *IEEE Transactions on Smart Grid* (8 5 2018). <https://doi.org/10.1109/TSG.2018.2834219>
- [23] Milad Moradi. 2016. A centralized reinforcement learning method for multi-agent job scheduling in Grid. *2016 6th International Conference on Computer and Knowledge Engineering (ICCCKE)* (2016), 171–176.
- [24] K. Moslehi and R. Kumar. 2010. A Reliability Perspective of the Smart Grid. *IEEE Transactions on Smart Grid* 1, 1 (June 2010), 57–64. <https://doi.org/10.1109/TSG.2010.2046346>
- [25] Ashutosh Nayak, Seokcheon Lee, and John W. Sutherland. 2018. Storage trade-offs and optimal load scheduling for cooperative consumers in a microgrid with different load types. *IJSE Transactions* 0, 0 (2018), 1–9. <https://doi.org/10.1080/24725854.2018.1460517>
- [26] D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra. 2010. Residential Demand Response Using Reinforcement Learning. In *2010 First IEEE International Conference on Smart Grid Communications*. 409–414. <https://doi.org/10.1109/SMARTGRID.2010.5622078>
- [27] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid. 2016. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. *IEEE Journal on Selected Areas in Communications* 34, 3 (March 2016), 510–527. <https://doi.org/10.1109/JSAC.2016.2525418>
- [28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.
- [29] Sachin Patil. 2013. Partially Observable Markov Decision Processes (POMDPs). <https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/pomdps.pdf>.
- [30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR abs/1707.06347* (2017). [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) <http://arxiv.org/abs/1707.06347>
- [31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550 (18 10 2017), 354 EP –. <https://doi.org/10.1038/nature24270>
- [32] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.