Energy Resiliency for a Large House through Intelligent Control using Reinforcement Learning

Ninad Gaikwad, and Anamika Dubey

***Abstract*—**

I. Introduction

A. Setup:

A community of N_h houses, in which N_{pv} houses have only rooftop PV installed, while $N_{pv,bat}$ houses have both rooftop PV and one Tesla Powerwall installed as energy storage. Hence, we have $N_h = N_{pv} + N_{pv,bat}$. The houses have to be powered through the combined PV+battery available in the community, to maintain habitable conditions during an extended outage after an extreme weather event like a hurricane.

B. Control Goals:

1) Maintain thermal comfort within the house by running the AC with the help of battery storage and PV.
2) Serve the total load demand (not including AC demand) for the house such that maximum load is served and at least critical load is always served.
3) Maintain the Tesla Powerwall SOC such that the system never runs out of energy till grid power is restored.

C. Flexibility for Intelligent Controller:

1) Flexibility in house total demand.
2) Flexibility due to thermal inertia of the house.

II. System Description and Models

The authors are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, Washington, USA. ninad.gaikwad@wsu.edu, and anamika.dubey@wsu.edu.

III. Control Algorithms

A. Reinforcement Learning-Based Controller

1) Markov Decision Process (MDP) Formulation: The proposed RL controller is designed to construct a control policy which can satisfy the goals mentioned in Section I. As a way to motivate RL design, we model the problem as a discrete-time Markov decision process comprised of the tuple $(\mathcal{X}, \mathcal{U}, \mathcal{P}, C, \beta)$, where $\mathcal{X}$ denotes the state-space, $\mathcal{U}$ denotes the action-space (a set of all *feasible* actions), $\mathcal{P}$ denotes the transition kernel, $C : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ denotes the cost function, and β denotes a discount factor. We denote by $X_k \in \mathcal{X}$ the state, and $U_k \in \mathcal{U}$ the control input at time-step k. Next, we define some of the key components of the MDP for our problem.

Choosing the state space: Based on the system described in Section II and the goals mentioned in Section I, it is natural to include the battery energy levels E_{bat} and the house internal temperatures T_h in the state X_k. In addition to these, we include certain exogenous inputs along with their forecasts, and previous time step's control input for the ACs for reasons discussed in the subsequent sections, as they can provide valuable information to the controller. So we have:

$$\begin{aligned} X_k := [&E_{bat}(k), T_h(k), \\ &T_{am}(k), \\ &E_l(k), E_{l,1}(k), E_{l,2}(k), E_{l,3}(k), \\ &E_{pv}(k), E_{pv,1}(k), E_{pv,2}(k), E_{pv,3}(k) \\ &u_{ac}(k-1)]^T \end{aligned}$$

Choosing the action space: Based on the system described in Section II, we define the control inputs as:

$$U_k := [u_{ac}(k), \Gamma(k), u_l(k)]^T, \tag{1}$$

where $\Gamma(k) \in \{-1, 0, 1\}$ denotes discharging, idle, charging at normal rate, and fast charging, respectively. The mapping of $\Gamma(k)$ into its constituent battery control commands $[c(k), d(k)]$ is defined below:

$$c(k) = \begin{cases} 1\,, & \text{if } \Gamma(k) = 1 \\ 0\,, & \text{otherwise} \end{cases} \tag{2}$$

$$d(k) = \begin{cases} 1\,, & \text{if } \Gamma(k) = -1 \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{3}$$

also, $u_{ac}(k) \in \{0, 1\}$ and $u_l(k) \in \mathbb{R}$

In total, there are 6 combinations of control inputs (U_k^i) possible for each house. So there are 6 feasible combination of control inputs for each house in total which are listed in

TABLE I: Feasible control inputs for each house U_k^i.

u_{ac}	0	0	0	1	1	1
Γ	-1	0	1	-1	0	1

Table I. Hence, for all N_h houses which all have pv installed and $N_h - N_{pv}$ houses which have pv+battery installed (N_{pv} is the number of houses which have just pv installed), we have $2^{N_h} \times 3^{N_h - N_{pv}}$ total different combinations for U_k.

Cost function design: The cost function is designed to capture four key features: (i) maintain the internal temperature of each house within bounds to provide thermal comfort to the occupants, (ii) turning on AC with high startup power consideration (iii) service the remaining loads of each house as desired by the occupants, while at least servicing the critical loads, (iv) service the remaining loads of each house in a fair manner and (v) keep the battery alive. The overall cost function is the sum:

$$C_{total} := C_h + C_{ac,on} + C_l + C_{bat} + C_{pv}, \tag{4}$$

where each of the terms are discussed in detail next.

The cost function for the house temperatures is:

$$C_h(X_k, U_k, X_{k+1}) = c_h + p_h\Big([T_h(k+1) - \bar{T}_h]_+ + [\underline{T}_h - T_h(k+1)]_+\Big) \tag{5}$$

where c_h is the (low) cost for maintaining T_h within bounds $[\underline{T}_h, \bar{T}_h]$, and p_h is the (large) penalty/cost for violating the bounds; see Figure **??**.

The cost function for turning on ACs with high startup power is:

$$C_{ac,on}(X_k, U_k, X_{k+1}) = \begin{cases} c_{ac,on}\ , & \text{if } u_{ac,on}(k)\bar{P}_{ac} \leq \\ & \Gamma_{discharging}(k)\bar{P}_{bat} + \\ & \dfrac{E_{pv}(k)}{\Delta T_s}, \\ p_{ac,on}\ , & \text{otherwise}\ . \end{cases} \tag{6}$$

where $p_{ac,on}$ is the (high) cost for not considering high startup power for the ACs, $c_{ac,on}$ is the (low) cost for considering high startup power for the ACs, where $u^i_{ac,on}$ and $\Gamma^i_{discharging}$ are the AC turn on and battery discharging variables respectively given as follows.

$$u_{ac,on}(k) = \begin{cases} 1\ , & \text{if } u_{ac}(k) = 1\ \&\ u_{ac}(k-1) = 0, \\ 0\ , & \text{otherwise}\ . \end{cases} \tag{7}$$

$$\Gamma_{discharging}(k) = \begin{cases} 1\ , & \text{if } \Gamma(k) = -1, \\ 0\ , & \text{otherwise}\ . \end{cases} \tag{8}$$

The cost function for the secondary load (lights-fans) is:

$$C_l(X_k, U_k, X_{k+1}) = c_l + p_l[\bar{E}_l(k) - E_l(k)]_+ \tag{9}$$

where p_l is the (high) cost for not servicing the remaining load when desired, c_l is the (low) cost for servicing when desired.

The cost function for maintaining the battery state of charge is:

$$C_{bat}(X_k, U_k, X_{k+1}) = c_{bat} + p_{bat}[\underline{E}_{bat} + \Delta E_{bat} - E_{bat}(k+1)]_+ \tag{10}$$

where c_{bat} is the (low) cost for keeping the battery alive, and p_{bat} is the (high) cost for allowing the battery to go below a prescribed minimum level of $\underline{E}_{bat} + \Delta E_{bat}$ and thus eventually die.

When there is no PV production potential (for example, during nighttime), charging the battery is not possible. Moreover, if a load is trying to be serviced when there is no PV potential, then the battery cannot be idle and needs to discharge. To discourage such undesired behaviors, we define the following cost function:

$$C_{pv}(X_k, U_k, X_{k+1}) = \begin{cases} p_{pv}\ , & \text{if } E_{pv}(k) = 0\ \& \\ & \Big(\Gamma(k) \in \{1\}\Big), \\ p_{pv}\ , & \text{if } E_{pv}(k) = 0\ \& \\ & \Big(u_{ac}(k) = 1\Big)\ \& \\ & \Big(\Gamma(k) \in \{0\}\Big), \\ 0\ , & \text{otherwise} \end{cases} \tag{11}$$

The constants c_h, p_h, $c_{ac_o n}$, $p_{ac,on}$, c_l, p_l, $c_{l,fair}$, $p_{l,fair}$, c_{bat}, p_{bat}, and p_{pv} are design choices.

2) Value Function Approximation and Zap Q-Learning: The goal is to obtain a state-feedback policy $\phi^* : \mathcal{X} \to \mathcal{U}$ that minimizes the sum of expected discounted cost:

$$\phi^* := \operatorname*{argmin}_{\phi:\mathcal{X}\to\mathcal{U}} \left\{ \sum_{k=0}^{\infty} \beta^k \mathsf{E}\big[c(X_k, U_k, X_{k+1})\big] \right\} \tag{12}$$

with $U_k = \phi(X_k)$ for $k \geq 0$.

Under the assumption that the underlying problem is an MDP, it is known that the optimal policy satisfies:

$$\phi^*(x) = \arg\min_{u\in\mathcal{U}(x)} Q^*(x, u)\,, \qquad x \in \mathcal{X} \tag{13}$$

where $Q^* : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ denotes the associated optimal Q function:

$$Q^*(x,u) := \min_{\{U_k\}} \sum_{k=0}^{\infty} \beta^k \mathsf{E}\big[c(X_k, U_k, X_{k+1}) | X_0 = x, U_0 = u\big]$$

where the minimization is over all feasible inputs.

Reinforcement learning algorithms such as Q-learning can be used to estimate an approximation for the Q-function. In this work, we use Zap Q-Learning to approximate Q^* using a parameterized family of functions $\{Q^\theta : \theta \in \mathbb{R}^d\}$ [**?**]. We employ a linear parameterization, so that,

$$Q^\theta(x, u) = f(x, u; \theta), \quad x \in \mathcal{X},\ u \in \mathcal{U}, \tag{14}$$

where $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a neural network parameterized by θ which approximates Q^*.

The size of θ depends on the architecture of the neural network employed and is a design choice. The input to the neural network is $[X_k]^T \in \mathbb{R}^{\mathbb{R}^{N(N_h+1)+3N_h-N_{pv}}}$, while the output is $Q^\theta(x,u) \in \mathbb{R}^{2^{N_h}\times 3^{N_h-N_{pv}}}$. Once the neural network is fixed, the Zap Q-learning algorithm can be used to estimate Q^* using the approximation Q^{θ^*}. We refer the interested reader to Algorithm 1 in [**?**] for details. The algorithm is implemented using the simulation models presented in Section **??**; given a current state X_k and a control input U_k, the state X_{k+1} at the next time step is obtained using these simulation models, and the tuple (X_k, U_k, X_{k+1}) is used to update the parameters θ.

3) Real-Time Control: The online state-feedback control is computed as follows:

$$\begin{aligned} U_k = \phi^{\theta^*}(X_k) &= \underset{u\in\mathcal{U}(X_k)}{\arg\min}\, Q^{\theta^*}(X_k, u) \\ &\approxeq \underset{u\in\mathcal{U}(X_k)}{\arg\min}\, f(X_k, u, \theta_T)\,,\ X_k \in \mathcal{X}, \end{aligned} \quad (15)$$

where θ_T is the estimate of θ^* obtained from the algorithm. *The minimum is over $2^{N_h} \times 3^{N_h-N_{pv}}$ values, so the optimization problem above is trivial to solve.*

B. Baseline Controller

1) Real-Time Control:

IV. Simulation Study Setup

The period selected for simulation is the time hurricane Irma passed over Gainesville, FL, USA, starting from its landfall on Sept. 11, 2017, to Sept. 17, 2017. Weather data is obtained from National Solar Radiation Database (nsrdb.nrel.gov). The simulations are run for 7 days starting at 00:00 hours (midnight) at day 1 (September 11, 2017) with a planning horizon of 24 hours and a time step of 10 minutes ($\Delta T_s = 10$ mins, $N = 144$) with battery initial state at $\bar{E}_{bat}$ (i.e., $E_{bat}(0) = \bar{E}_{bat}$) and the refrigerator initial temperature at $2^{\circ}C$ (i.e., $T_{fr}(0) = 2^{\circ}C$). The internal house temperature, $T_{house}(k)$, for the planning horizon is computed using the linear model given by [**?**], which models a typical, detached, two-story house in the USA.

A. PV Battery System Sizing

B. Computation

The plant is simulated in MATLAB. The optimization problem is solved using GUROBI [**?**], a mixed integer linear programming solver, on a Desktop Linux computer with 8GB RAM and a 3.60 GHz $\times$ 8 CPU.

C. Simulation Parameters

The parameters for the plant components; PV panels: ; Battery: and Loads: AC -. The system voltage is $V_d = 24\ V$ and the inverter efficiency is $\eta_{inv} = 0.9$.

The parameters for the house thermal model are .

The parameters for the optimization problem are .

V. Results and Discussion

VI. Conclusion