

# Keeping the Lights On for Campus

Using Deep Q-Learning and MDPs for Discrete and Continuous Micro-Energy Grid Optimization

Yasmina Abukhadra, Aakash Mishra

yasabukh@stanford.edu, aamishra@stanford.edu

## Abstract

Micro-grids such as college campuses with sustainable energy sources and non-renewable mixtures present an optimization opportunity for cost-saving, sustainability efforts and energy conservation. By using the demand, generation, and storage data from these micro-grids we demonstrate that we can design a Markov-Decision Process-based control system that can take real-time data for each grid and find the optimal allocation of resources to reduce non-renewable usage and buying energy from larger energy providers while also maximizing meeting demand constraints. We find that by using deep Q-learning with a discrete action space, our learned policy can save university campuses upwards of \$50,000 over greedy policies.

## Introduction

**Stochasticity in Microgrids.** Microgrid optimization is an emerging field that involves the optimization of renewable energy sources as well as the allocation of energy storage and demand requirements (Diddigi, Reddy, and Bhatnagar 2017). Many aspects of this problem are sequential in nature due to the constantly fluctuating energy demand and generation that depend on previous states. Often times, renewable energy sources fluctuate in generation, affecting the overall efficiency of the grid. This must be taken into account by using battery storage to prepare for low generation gap periods (Raju, Sankar, and Milton 2015). Similar stochastic behaviors can be observed in other types of renewable energy sources such as wind and hydroelectric sources (Srinivasan et al. 2023).

**Markov Decision Processes & Q-Learning.** Given the randomness in the generation of renewables and demand structure in microgrids, the problem space can be structured as a markov-decision process with certain assumptions. We assume the cost  $R(s, a)$  and the state-action space transition function  $T(s, a)$  are dependent on the current state only (Puterman 2014). Given the large size of the state action space  $|S||A|$  we plan on utilizing Q-learning with gradient ascent to develop an effective policy. Each time interval can be discretized in order to define a set of states for each training iteration to consider (Jain et al. 2018).

**Applications for University Campus Grids.** The goal of our research is to find more energy-efficient solutions for a university microgrid, with insights that could potentially help the Stanford campus microgrid. Stanford generates a large portion of its own energy and pulls power from the grid to meet its remaining energy requirements (ses ). Daily energy use on university campuses varies widely, depending on the occurrence of sports games, the use of study rooms and the amount of energy used for building temperature regulation. Using reinforcement learning for microgrid optimization, especially with renewable energy sources can help the university with cost-cutting and also help it meet its energy goals (R. Singh et al. 2024). While the energy generation and usage data from Stanford's campus are not openly available we can use UC San Diego as a suitable proxy (Silwal et al. 2021).

**Contributions.** We demonstrate a parameterized Q-learning policy approach as well as a deep learning policy approach that achieves monetary savings over a random baseline. We provide insights into greedy heuristics that can help with energy investment decisions and trade-offs with differing amounts of battery storage infrastructure.

## Related Work

**Markov Models for Renewable Energy.** Several Markov-based models have been proposed in the past to model the energy trade-offs of renewables. This includes Markov models for wind-based energy grids and photovoltaic grids as well (Chen et al. 2009). Then many papers expanded on this with 24-hour horizon rollouts for short-term generation modeling with Markov chains (Pesch et al. 2015).

**Markov-based Grid Optimization** With the introduction of an observability model, previous papers have created Partially Observed Markov Decision Process-based (POMDPs) models to create policies for the energy grid. The use of the observability model is highly dependent on the grid and for our research project we could not make the same assumptions about the data and max-capping of demand or generation (Durante, Nascimento, and Powell 2017). Other methods use deep-learning with linear programming to satisfy constraints for meeting demand while maximizing renewable usage (Srinivasan et al. 2023). Future implementations built on q-learning methods by using forecasting models that could predict renewable generation more effectively

and incorporated these features into the feature vector for the POMDP.

**Microgrid Applications.** There are several studies that completed working examples of MDP controllers for smaller scale energy grids. These studies used methods ranging from linear programming to deep Q-learning to train MDP controllers to optimize microgrid environments (Henri et al. 2020). Recently, reinforcement learning has been successfully used to control microgrids with battery model uncertainty, showing improvements over model predictive control and rule based control approaches (Halev, Liu, and Liu 2024). Fully-decentralized multiagent reinforcement learning has also been used to successfully balance state of charge in a distributed energy storage system within a microgrid (Xiong et al. 2024).

## Methods

### Simple Battery Storage Model

We first tried to train an MDP controller to optimize for the amount of energy to transfer in and out of the battery given the current demand and generation conditions from the UCSD dataset. We wanted to try out this simple example before trying to optimize every action type in the grid. We defined our state as follows:

$$s_t = \{g_t, r_t, b_t\}$$

- $g_t$ : generation at time  $t$ ,
- $r_t$ : renewable production at time  $t$ ,
- $b_t$ : battery storage at time  $t$ .

We create a feature vector  $\phi$  that stored the different features based on time, interaction terms between generation, production and battery storage amounts as well as the action taken as follows:

$$\phi(s_t, a_t) = \{t, g_t, r_t, b_t, g_t \cdot r_t, g_t \cdot b_t, r_t \cdot b_t, a_t\}$$

We used two different formulations of the q function, one that was a simple parameterized q-function for learning and another that involved deep-learning to estimate the q value.

$$Q_\theta(s_t, a_t) = \theta^\top \phi(s_t, a_t)$$

$$Q_{NN}(s_t, a_t) = f_{NN}(s_t, a_t; \theta)$$

Finally we constrained the amount of energy that could be used to fill the battery by the total amount of over generation.

$$\text{storage}_t = \max(0, d_t - g_t)$$

We also defined the constraint for the amount of energy that could be used from the battery as well.

$$\text{usage}_t = \min(d_t - g_t, b_t, a_t)$$

For our reward, we specified a penalty that was dependent on the gap between the generation + battery storage versus demand.

$$\text{Penalty}_{\text{gap}} = \lambda_1 \cdot |d_t - g_t - b_t|$$

And we also penalized the amount of either action when the model tried to go past the bounds of what was possible to allocate as well.

$$\text{Penalty}_{\text{action}} = \lambda_2 \cdot \left( I(a_t > \text{storage}_t) + I(a_t < \text{usage}_t) \right)$$

Combined together our final objective was the summation of these two penalty terms for our reward.

$$R_t = -(\text{Penalty}_{\text{gap}} + \text{Penalty}_{\text{action}})$$

We then trained the models for 20 episodes each with a learning rate of 0.01 and an epsilon value of 0.1 and compared their results to an epsilon greedy baseline and a random action baseline. We defined the deep Q-learning regressor as follows:

Listing 1: Definition of the QNetwork class in PyTorch

```
import torch
import torch.nn as nn

class QNetwork(nn.Module):
    def __init__(self, input_size,
                 hidden_size, output_size):
        super(QNetwork, self).__init__()
        self.fc1 = nn.Linear(input_size,
                              hidden_size)
        self.fc2 = nn.Linear(hidden_size,
                              hidden_size)
        self.fc3 = nn.Linear(hidden_size,
                              output_size)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return self.fc3(x)
```

We then used Adam to optimize the parameters and mean-squared error for our loss criterion.

### Pymgrid Model

In order to simulate power consumption and usage for the microgrid we are trying to optimize over, we use pymgrid (Henri et al. 2020), a python library for creating and simulating microgrids. We create two battery modules, a renewable module to represent the energy generated on-campus, a grid module to represent the ability to import energy from San Diego Gas and Electric, and a load module to represent the energy demand. In order to do this we assume trends and data distributions similar to (Silwal et al. 2021).

We import UCSD’s generation and load data into the renewable module and load module in order to create a simulator based on the energy usage and generation trends present on the UCSD campus. Additionally, we simulate the dynamic cost of electricity by using information about SDG&E pricing. The timing of pricing for small commercial customers can be found on the SD&E website (SDG&E b). For

| Benchmark | #actions | #modules | #episodes | random    | epsilon-greedy (0.1) | neural network | greedy vs. NN reward |
|-----------|----------|----------|-----------|-----------|----------------------|----------------|----------------------|
| 1         | 12       | 6        | 3         | -1.40e+08 | -7.27e+07            | -6.16e+07      | -1.11e+07            |
| 2         | 4        | 5        | 5         | -5.56e+08 | -1.21e+08            | -7.40e+07      | -4.73e+07            |
| 3         | 4        | 5        | 5         | -8.86e+08 | -1.75e+08            | -1.13e+08      | -6.15e+07            |
| 5         | 4        | 5        | 5         | -4.23e+08 | -8.33e+07            | -5.27e+07      | -3.06e+07            |
| 7         | 4        | 5        | 5         | -9.91e+07 | -1.88e+07            | -1.25e+07      | -6.32e+06            |
| 15        | 4        | 5        | 5         | -2.54e+08 | -5.22e+07            | -3.46e+07      | -1.77e+07            |
| 17        | 4        | 5        | 5         | -4.58e+08 | -8.87e+07            | -5.79e+07      | -5.19e+07            |
| 19        | 4        | 5        | 5         | -1.90e+07 | -3.89e+06            | -2.19e+06      | -1.70e+06            |
| 20        | 4        | 5        | 5         | -4.65e+08 | -1.06e+08            | -6.33e+07      | -4.23e+07            |
| 23        | 4        | 5        | 5         | -1.50e+08 | -3.19e+07            | -2.30e+07      | -8.88e+06            |
| UCSD      | 6        | 6        | 5         | -1.19e+08 | -1.19e+08            | -1.19e+08      | -5.18e+07            |

Table 1: Top-10 Best Deep-Q-Learning Microgrid Benchmark Results and the comparison on UCSD data.

simplicity, we assume the weekday peak timing structure, where the peak cost is between 4pm and 9pm, the off peak is between 6am-4pm and 9pm-12am, and the super off peak cost is 12am-6am. The months of June through October have higher price points than the rest of the year. We used the cost information for the season and peak/off-peak/super off-peak categories from 2018 (SDG&E a).

The battery modules created approximate the batteries present in the UCSD system, with one battery having a maximum capacity of 5kW and set to have a maximum usage of 2.5kW per step and another battery having a maximum capacity of 200kW and set to have maximum usage of 100kW per step. Both batteries are initialized to a 50% state of charge at the start of the simulation and are assumed to have 90% efficiency (this is just an assumption, not based on UCSD battery specs).

The simulation environment is created from the microgrid using `pymgrid.envs.DiscreteMicrogridEnv`. It uses a discrete action space, and allows us to full observe the current state of the system, as well as receive rewards which are based on the cost of the action, according to the energy pricing. In this environment, any renewable energy available is automatically used towards fulfilling the demand and the action then specifies a priority list from which to deploy each of the batteries versus the grid in order to meet the remaining energy demand. Effectively, at each step we choose whether to draw from one of the batteries first or the grid. In addition to the simulation performed with UCSD microgrid data, we used the 25 benchmark grids from PymGrid in order to test the performance of deep-q-learning on a wider variety of microgrids and datasets.

Similar to that described above in Listing 1, we used a QNetwork, Adam optimization, and mean-squared error in order to perform deep-q-learning on these simulated microgrid environments. One episode of training constituted stepping through the entire simulation. Exploration was favored over exploitation 20% of the time during training. After training for the desired number of episodes, inference was run using the trained model in order to step through the entire simulation a final time.

We compared the results of this deep-Q-learning process against a random action baseline, where the action (priority

list) chosen at each step is random, as well as an epsilon-greedy baseline with an epsilon value of 0.1.

## Results

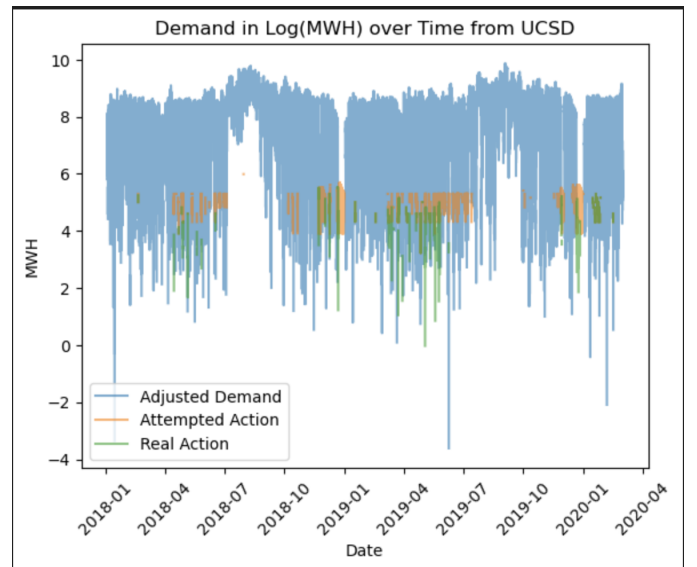


Figure 1: Simple Battery Model Attempted and Real Actions for the UCSD Dataset 2018-2020

**Simple Battery Storage Model.** For the continuous form of this problem we found that the model had a difficult time trying to learn the precise values of energy to allocate to the battery in the grid. That made it difficult for the model to perform much better than the random or epsilon greedy baseline. Even after we added the secondary penalty term, we found that the model still had a hard time adjusting to realistic values for deciding the amount to allocate. The other difficult of the UCSD dataset is that the battery is too small and the over generation periods are too short. This means that the times where an action can be taken regarding the battery is short-lived and hence difficult to train on. As seen in figure 1, there are few periods where the model is actually

able to accomplish an action.

**Discrete PymGrid Optimization.** We ran the deep Q-learning method on all 25 of the benchmarks in the PymGrid repository and compared that to a random baseline as well as a epsilon greedy baseline where  $\epsilon = 0.1$ . We found that the deep Q-learning policy after 2 to 5 episodes performed better than both the greedy and the random baseline. We also found that the Q-learning policy did better than random and greedy on the UCSD microgrid baseline as well although the differential was not very high. We showcase the best 10 benchmark results as well as the UCSD result in table 1.

Note that in this table the greedy vs. NN column is calculated by taking the difference between the greedy reward and the Q-learning reward. The more negative that column's value is, the better the deep-learning policy did. We find that the policy struggled with action spaces on the extremes of high-order and low-order, namely the microgrids with 2 or 12 actions. If the number of modules went up to 6 instead of 5, then oftentimes we saw that the Q-learning struggled more to improve over the baseline.

Figures 2 through 4 display the cumulative costs accrued in the simulation when following each of the benchmark policies versus the deep-q-learning policy. The neural network was effective in reducing the cost from the unbalanced energy module, representing the cost in an imbalance between the generated and demanded energy. The neural network also effectively reduced the overall cost as seen in the balance module. We can see in Figure 4 that the neural network learned to favor the grid over the genset exclusively. We see that for many of the microgrid benchmarks, the random and greedy policies struggle with keeping the diverse set of penalties down whereas the neural network based deep Q-learning policy mainly struggles with the difficulty of the balance penalty module. For a graph, the more visible linear lines, the more diverse the action choices were for that policy. We see that the deep-Q policy tended to have a much more limited selections of actions that it favored as compared to random and greedy. Generally we also see that while all of the policies tend to have some sort of linearly decrease cumulative rewards trend, the magnitudes differ when it comes to actual loss with the NN MDP having the least.

## Discussion

The ability of the deep-Q learning MDP to quickly learn policies that were much better than the greedy baseline showcases the effectiveness of the method in diverse use case. We had also tried to attempt a parametrized version for both the simple battery model as well as the discrete PymGrid space, but this attempt did not yield any prominent results. We found that the parametrized approach struggled with being able to prioritize more than one action and was very limited in-terms of the Q-value approximation. When observing the the parameterized model outputs, we noticed that the Q-value between the different 1-step horizon options were very similar and often random noise due to gradient updates would cause the training actions to vary heavily. As an implementation decision, we chose to generate full results for the deep-learning approach which successfully beat the

greedy baseline with only a few episodes of training at a time.

The convergence of a MDP policy also depended on the actual actionable space. In many cases, because the period of over generation was low, there were very few times the model could actually be able to prioritize saving energy to the battery. Hence, we see greater performance gains for MDP controllers when the limitations on the action space are smaller.

When observing the figures for Microgrids 2, 3 and 13, we also see that the trends in the random policy and greedy policy generally tend to indicate more penalties from unbalanced energy and less consistent use or preference of one source of energy. The neural network learns to prioritize the energy source that minimizes overall cost as well as reduces the penalties from unmet energy needs.

## Future Work

The discrete action space within PymGrid limited the ability of the MDP controller to determine the amount of energy flow, as the action space only allows for prioritizing the deployment of modules in a set manner. In future work, one area of interest would be to see what could happen if the model that was allowed to draw more energy from the grid than necessary during off-peak hours in order to be prepared for demand during peak hours. The UCSD dataset did not include many instances where generation outweighed demand, so it was difficult to learn a policy for when batteries should be used, as they often did not have the opportunity to charge enough to be used effectively. PymGrid's discrete environment also does not allow the opportunity to store renewable energy instead of using it towards current load; in cases where the cost of grid energy is low, it might make sense to store the effectively zero cost solar energy at a later time when the cost of grid energy is high. It would be interesting to explore a dataset and simulation environment with more opportunities for battery charging and energy storage in order to learn a more insightful policy for battery usage.

While we used the UCSD dataset as a proxy to show that we can try to optimize the energy usage on campus, we also have more granular data about each building's usage and each power plant's generation. For a higher fidelity proxy of Stanford, we can map the buildings and generation plants that are similar to Stanford and then use that to come up with a more accurate demand and generation schedule. Reinforcement learning has previously been used in order to optimize energy use taking into account electric vehicle charging and HVAC system control (Chen et al. 2022). Taking these factors into account would also be interesting, especially on a building-level scale. The UCSD dataset included EV charging data, so that would be a logical next-step to explore (Silwal et al. 2021).

Another area of interest for this project could be demand and generation forecasting. By using some form of Long-Short Term Networks (LSTMs) we can use information about the weather and other features to predict what the generation and demand would look like a few steps ahead. We can then use this as a part of our feature vector to help with learning an effective policy.

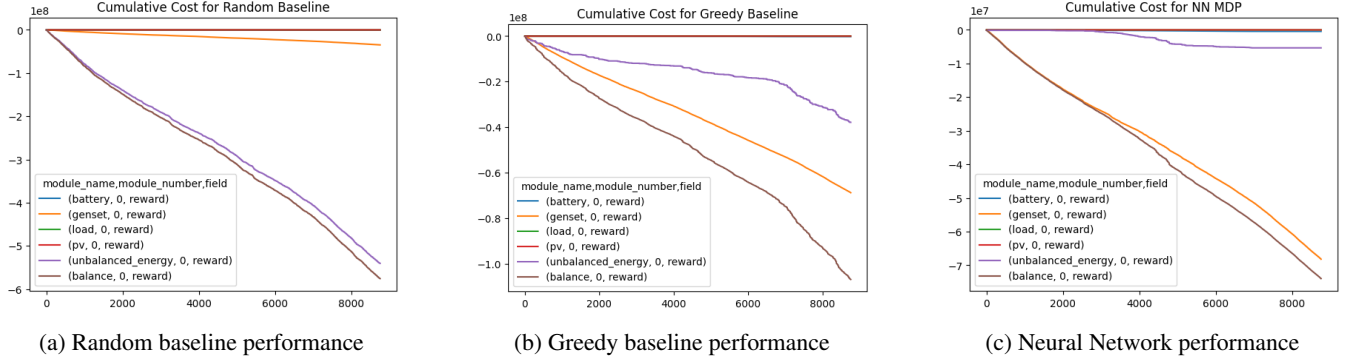


Figure 2: Cumulative cost plots for grid 2. The reward for each module represents the monetary cost associated with the module and action taken at that step. For the pv module costs are often zero. For the battery module, there can be some small cost to charging. The load module represents the energy demands of the system and tends to have no associated reward. The unbalanced energy reward represents the cost of over- or under-generation and the balance reward represents the overall cost.

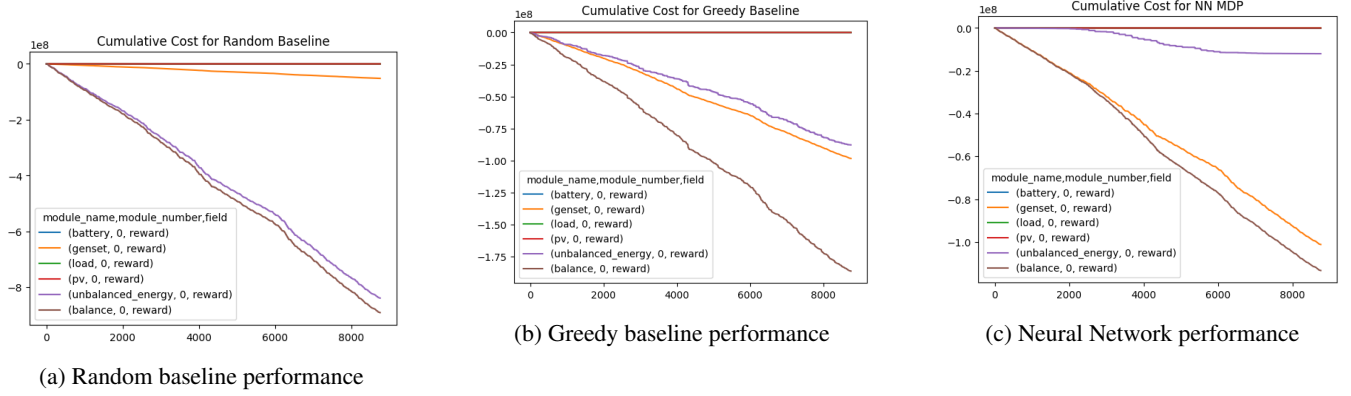


Figure 3: Cumulative cost plots for grid 3.

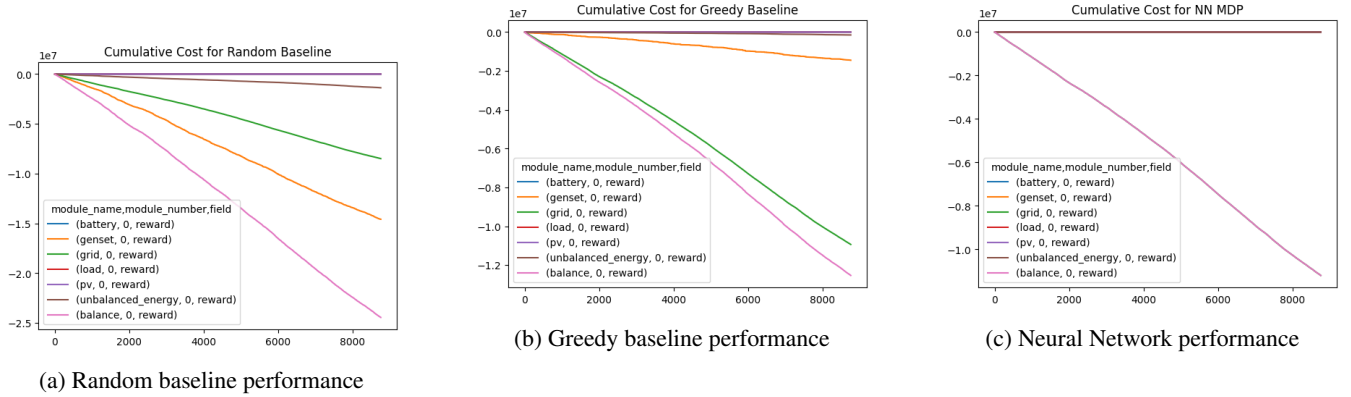


Figure 4: Cumulative cost plots for grid 13.

Lastly, given the heavy emphasis on trade-offs between renewables and non-renewable sources of energy, we can also factor in penalty costs for using the greater grid given that the mixture of energy sources from companies like Palo Alto utilities will often have more usage of non-renewables and greater production of CO<sub>2</sub>. Ideally, we could gain access to the current optimization that the Stanford Microgrid does and the data from the office of sustainability to determine a more rigorous policy that can be used by the university.

## References

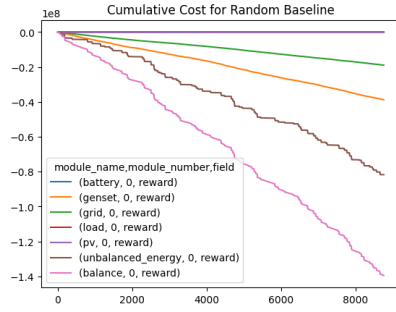
- Chen, P.; Berthelsen, K. K.; Bak-Jensen, B.; and Chen, Z. 2009. Markov model of wind power time series using bayesian inference of transition matrix. In *2009 35th Annual Conference of IEEE Industrial Electronics*, 627–632. IEEE.
- Chen, X.; Qu, G.; Tang, Y.; Low, S.; and Li, N. 2022. Reinforcement learning for selective key applications in power systems: Recent advances and future challenges. *IEEE Transactions on Smart Grid* 13(4):2935–2958.
- Diddigi, R. B.; Reddy, D.; and Bhatnagar, S. 2017. Multi-agent q-learning for minimizing demand-supply power deficit in microgrids. *arXiv preprint arXiv:1708.07732*.
- Durante, J. L.; Nascimento, J.; and Powell, W. B. 2017. Backward approximate dynamic programming with hidden semi-markov stochastic models in energy storage optimization. *arXiv preprint arXiv:1710.03914*.
- Halev, A.; Liu, Y.; and Liu, X. 2024. Microgrid control under uncertainty. *Engineering Applications of Artificial Intelligence* 138:109360.
- Henri, G.; Levent, T.; Halev, A.; Alami, R.; and Cordier, P. 2020. pymgrid: An open-source python microgrid simulator for applied artificial intelligence research. *arXiv preprint arXiv:2011.08004*.
- Jain, P.; Mueller, J. A.; Park, J.; Landers, R.; and Kimball, J. W. 2018. Battery optimization in microgrids using markov decision process integrated with load and solar forecasting. In *2018 9th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, 1–8. IEEE.
- Pesch, T.; Schröders, S.; Allelein, H.-J.; and Hake, J.-F. 2015. A new markov-chain-related statistical approach for modelling synthetic wind power time series. *New journal of physics* 17(5):055001.
- Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
- R. Singh, A.; Kumar, R. S.; Bajaj, M.; Khadse, C. B.; and Zaitsev, I. 2024. Machine learning-based energy management and power forecasting in grid-connected microgrids with multiple distributed energy sources. *Scientific Reports* 14(1):19207.
- Raju, L.; Sankar, S.; and Milton, R. 2015. Distributed optimization of solar micro-grid using multi agent reinforcement learning. *Procedia Computer Science* 46:231–239.
- SDG&E. Available rates for small commercial customers (effective 1/1/18). [https://www.sdge.com/sites/default/files/regulatory/SummarySDG&E. Total electric rates — san diego gas electric.](https://www.sdge.com/sites/default/files/regulatory/SummarySDG&E.Total%20electric%20rates%20-%20san%20diego%20gas%20electric.pdf) <https://www.sdge.com/total-electric-rates>.
- Stanford Energy Systems Innovations (SESI): Renewable Energy. <https://sesi.stanford.edu/renewable-energy>.
- Silwal, S.; Mullican, C.; Chen, Y.-A.; Ghosh, A.; Dilliot, J.; and Kleissl, J. 2021. Open-source multi-year power generation, consumption, and storage data in a microgrid. *Journal of Renewable and Sustainable Energy* 13(2):025301.
- Srinivasan, S.; Kumarasamy, S.; Andreadakis, Z. E.; and Lind, P. G. 2023. Artificial intelligence and mathematical models of power grids driven by renewable energy sources: A survey. *Energies* 16(14):5383.
- Xiong, Z.; Luo, B.; Wang, B.-C.; Xu, X.; Liu, X.; and Huang, T. 2024. Decentralized multiagent reinforcement learning based state-of-charge balancing strategy for distributed energy storage system. *IEEE Transactions on Industrial Informatics* 20(10):12450–12460.

## Appendix

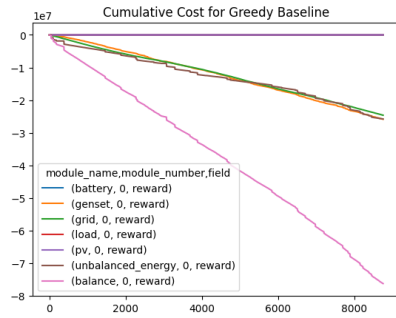
### Group Member Contributions

**Aakash:** battery model, framework for training and testing, running analysis, figures, and report

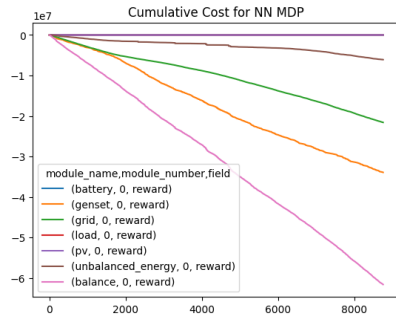
**Yasmina:** pymgrid-based model, cost data for UCSD, running analysis, figures, and report



(a) Random baseline performance



(b) Greedy baseline performance



(c) Neural Network performance

Figure 5: Cumulative cost plots for grid 1.