Reinforcement Learning for Simple Microgrid Control

CS 138 Final Project

Joseph Egan

Abstract

In this paper, I introduce the concept of a microgrid and how reinforcement learning is an advantageous method for controlling its operations. Two similar microgrid environments are proposed, and an agent utilizing deep q-learning is described. The agent has the ability to control the price of power to consumers, the utilization of an energy storage system, and the interactions between the microgrid and the main grid. Ultimately, it is shown that the agent is able to learn the correct control policies to effectively manage the microgrid environment to maximize reward. Finally, ideas for further exploration into this topic are proposed.

1 Introduction

Centralized electricity distribution is the most common method of supplying electricity to communities in most developed countries. This method is constituted by what we know of as the "grid" -- networks of power and transmission lines that carry electricity from large generation facilities, through substations, and then finally to homes and businesses.

As we have seen in recent years, this model of distribution has many significant drawbacks. Inclement weather can cause severe disruptions in service and sharp increases in prices in not only the affected but surrounding areas. Distributing power across long distances is less efficient and more expensive – requiring high-voltage transmission lines that span hundreds of miles and which must be rigorously maintained. Additionally, the very nature of centralized energy generation, which must supply consistent power to large swaths of communities, advantages non-renewable sources of energy, due to their ability to consistently produce power (fossil fuel burning 24/7 vs solar power only generated during the day). Finally, large networks of infrastructure on which many communities rely are exceedingly vulnerable to cyber or physical terrorism and attack.

In contrast, microgrids are a decentralized method of supplying communities with electricity. They are simply local electrical grids capable of supplying all or nearly all of the load demands of their community. A microgrid in its simplest form consists of a source of local power generation (wind, solar, natural gas, etc.), consumers (households, businesses, etc.), and a reservoir of energy storage such as a battery. They can either be isolated or connected to a 'main' grid through a PCC (point of common coupling).

Microgrids can alleviate many of the above-stated problems caused by centralized distribution but are not without their own issues. Smaller scale can lead to higher volatility in demand and also generation, as renewable sources can be intermittent.

Controlling and optimizing the various processes of the local grid can be complicated using conventional model-based approaches. Typically, a high level of expertise is needed to tune a control model to fit a specific microgrid situation and changing features can often result in an entire revamp of the control process. As a result, these models can be expensive and are not scalable. Given that microgrids have the most potential to be useful in fringe or developing communities, this can be a high barrier to entry.

Reinforcement learning is a potential solution to this problem. Since the algorithms do not require a model of the environment, they are able to approximate optimal control strategies from operational and environmental data. In this paper, I demonstrate that a simple reinforcement learning algorithm is able to learn near optimal policies for managing a microgrid environment generated from real weather and electrical data from the Boston area.

2 Background Related Work

Microgrid control has been studied extensively by the reinforcement learning community. One of the main challenges is to manage the trade-offs between various objectives, such as minimizing the cost of energy, maximizing the use of renewable energy sources, and maintaining a stable and reliable power supply. The existence of multiple objectives has led to numerous papers on the optimization of specific features such as cost or efficiency, or comparisons across different microgrid attributes.

One popular sub-domain centers around maximizing utilization of local distributed energy resources and minimizing dependency on the main grid. In these problems, the agent is taught to optimize the scheduling of charging or discharging of energy storage options, and/or activating additional energy generating resources. In [1], it was shown that Q-learning via a convolutional neural network could learn to optimize scheduling for a battery, hydrogen storage, and a diesel backup generator.

Another sub-domain broadens the control problem by incentivizing the agent to simply optimize gross margin (revenue – cost of generation). [2] provides an in-depth analysis of several reinforcement learning algorithms in one of the most complex microgrid environments proposed. While the energy resource was simply a wind farm, they created two different load components – thermostatically controlled load (temperature dependent) and price sensitive load (price dependent). This allowed them to implement real weather and pricing data into their simulation (from Finland) and show that a variety of deep reinforcement learning algorithms (DQN, Double DQN, SARSA, Actor-Critic, and more) were able to find near-optimal policies. The agents were shown to be able to manage a broad action space consisting of load control, price mechanics, battery charging, and interactions with the main grid.

In [3], DQN is also used, but in an environment with multiple renewable energy sources (wind, solar, micro-turbine, and hydrogen fuel cells). Similar to [2] They utilize a deep feedforward neural network to approximate the values of the state-action pairs. However, instead of optimizing gross margin, they simply try to minimize operating costs. They also found that the algorithm was able to effectively schedule charging and discharging along with selling and purchasing with the main grid to minimize costs.

In this project, I aimed to apply some of these concepts to a novel dataset based on the Boston area without using any of the traditional reinforcement learning python packages such as OpenAI Gym.

3 Technical Approach

I explored two simple problems commonly explored within microgrid research. In problem 1, reinforcement learning is applied to show that it can be used to optimally charge and discharge the battery within a microgrid. In problem 2, reinforcement learning is applied to show that an agent can optimize battery control in addition to pricing mechanics in order to maximize revenue.

To begin, the problems were framed as a Markov-Decision Process (MDP). An MDP consists of possible states, actions that can be taken within those states, a transition function that determines the movement between states based on the environment and actions taken by the agent, and finally a function that determines what reward, if any, is earned upon the agent taking an action. The agent learns to optimally interact with the environment through the pursuit of the maximization of the rewards received.

Environment

The environment consisted of a small microgrid powered by both wind and solar renewable energy sources. It also contained a battery capable of storing 2 hours' worth of average load. Finally, the microgrid was allowed to interact with a main grid at disadvantageous hourly prices for the microgrid. At each hour, values for each of these components were generated from average weather and electrical data from the Boston area [4][5], plus some small noise, and scaled to a town of 10,000 residents.

States

The state space in both problems consisted of the statuses of the many components of the microgrid environment. Additionally, I assume that the agent could generate accurate forecasts of many of these features up to 4 hours into the future. These features were combined at each step to generate a vector with 13 elements in problem 1, and 14 elements in problem 2:

Battery State of Charge: The agent was given the current power availability of the battery.
Power Generation: The agent received 4 values depicting the microgrids generation in the current hour, plus the next three hours.
Power Demand: Similarly, the agent received 4 values depicting the amount of power requested by the 10,000 residents in the current hour plus the next three hours.
Pricing: The agent also received the real-time price of electricity plus three hours of forecast from the main grid.
Consumer Banked Load: In problem 2, the agent also received a state feature that depicted the amount of excess load that had been deferred from previous hours due to the price actions of the agent. For example, if the agent chose to raise the price, consumers with elastic demand would reduce energy consumption, but in the future would still need to consume that power eventually.

Actions

The action space consisted of an excess energy action, deficient energy action, and a price action:

Excess Energy: in the event where power generated exceeded demand, the agent could specify whether to use the excess to charge the battery or to sell the power to the main grid. Power remaining after charging the battery was automatically sold.
Deficient Energy: if power demanded exceeded power generated, the agent could specify whether to discharge from the battery or to buy from the main grid. Any power deficit remaining after discharging the battery was satisfied by purchasing power from the main grid.
Price Action: In problem 2, the agent could specify one of three prices movements: [-1, 0 ,1] corresponding to a 15% decrease, no change, or 15% increase in prices respectively. This has the effect of increasing or decreasing power demanded at that time step based on an elastic demand function.

Rewards

In problem 1, since the agent had no control over the price, it only received reward equal to the cost of energy supplied from the main grid. This was scaled to max out just below 0 to replicate the concept of minimizing dependence on the main grid.
In problem 2, the agent received the same reward from problem 1 in addition to the amount of energy generated and delivered to residents, multiplied by the price it set at that time step. This tended to be somewhere between $12,000 and $14,000 per day. For both problems, reward was delayed across all time steps until the very last time step in the episode (final hour of the day) in which it received the sum of all rewards earned in that episode. This was done to try to reduce the likelihood of the

agent selecting actions that while leading to high reward in that time step, ultimately lead to worse overall outcomes in the episode.

Agent

Q learning

The agent in this project used Q-learning to learn how to navigate the microgrid environment. In Q-learning, the agent formulates an estimated value for each state-action pair it encounters. Each time the agent encounters a state-action pair, it updates the estimated value of the pair based on the reward it receives plus the estimated value of the best action in the next state, scaled by a discount factor and learning rate. Q-learning was used in this project for its simplicity and high interpretability.

Deep Q Learning

In this project, each element of the state vector is continuous. As a result, it is impossible to assign values to every possible state-action pair. Therefore, the agent instead trains a neural network where the input is the state feature vector, and the output is a value for each possible action.

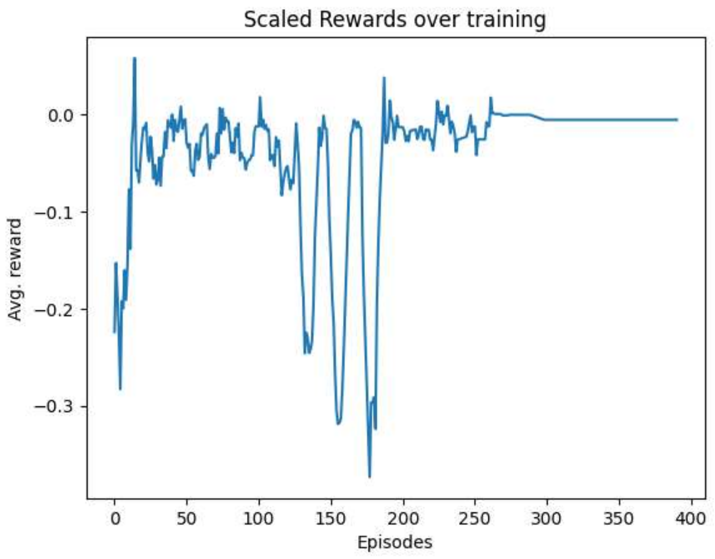
The agent generates training data for the network by interacting with the environment. At each step, it adds the state, action, reward, and next state to a memory queue. When the memory queue reaches a sufficient size, the agent samples the queue for training experiences. The true values of the actions used for estimating the loss of the network are generated from a second neural network, whose parameters are fixed but periodically updated to match the main network. The agent uses epsilon-greedy exploration. At each time step, a random number between 0 and 1 is generated. If the random number is below epsilon, a random action is taken. If the random number is greater than epsilon, then the agent performs what it believes to be the most valuable action in that state. Over time, epsilon is decayed so that the agent explores random actions less-frequently.

Agent-Environment Interaction

Each episode is one day, consisting of 24 time-steps which are hours. The environment generates a state vector which is delivered to the agent. In exchange, the environment receives either a random action or the agent’s best action, consisting of the excess energy action, deficient energy action, and, in problem 2, also a price action. Following these actions, the battery is either charged or discharged, power is bought or sold, and the price of power goes up, down, or stays flat. The result of these actions leads to a new state which is turned into a vector and returned back to the agent along with any reward.

4 Evaluation of Results

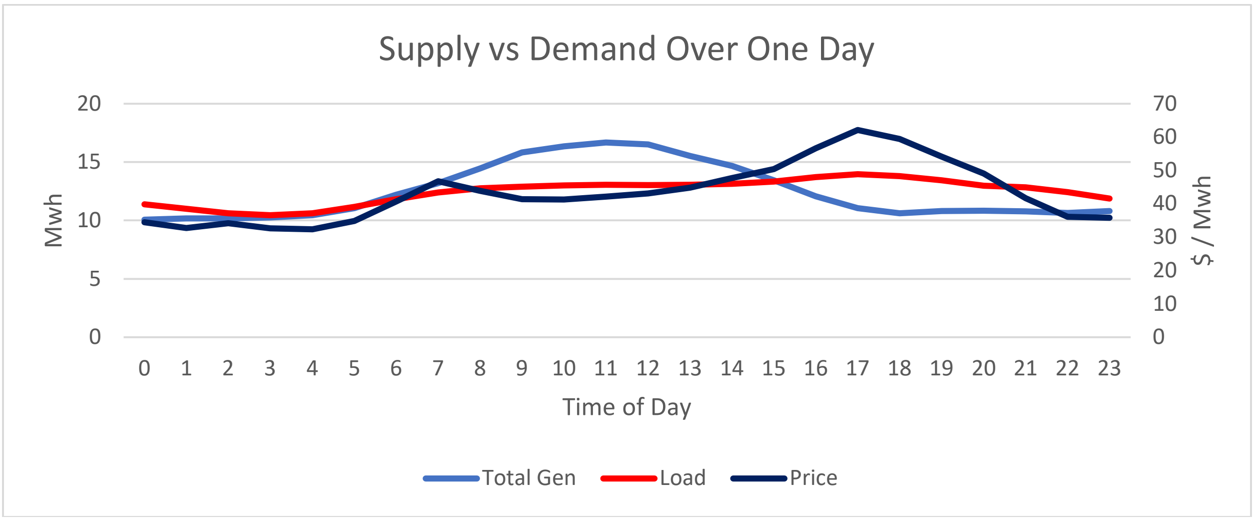
Both problems were evaluated with an initial epsilon of 0.15 which was decayed linearly. In problem 1, epsilon was set to 0 at episode 300, and the agent thereafter only performed it's estimated optimal actions:



rolling 10-day average of rewards (to smooth curve in the absence of the ability to run multiple iterations)

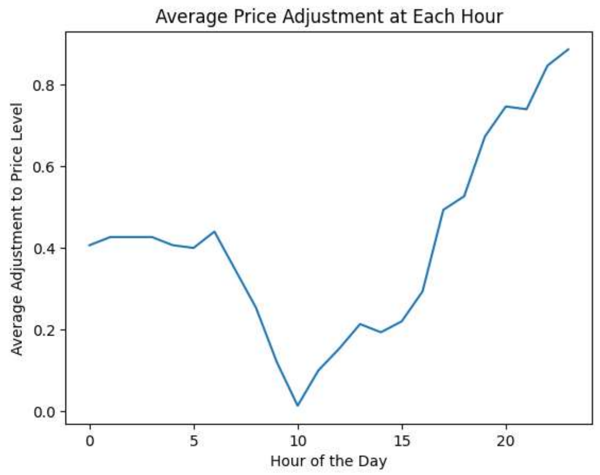
Ideally, I could have run this agent several times on different seeds, but more than one iteration would crash my computer. I believe that the instability of the reward between 125 and 175 episodes is due to the replay memory of the agent. After 125 episodes, it began dropping stored transitions. However, problem 1 was a clear proof of concept that the agent could learn the simplest level of microgrid management tasks.

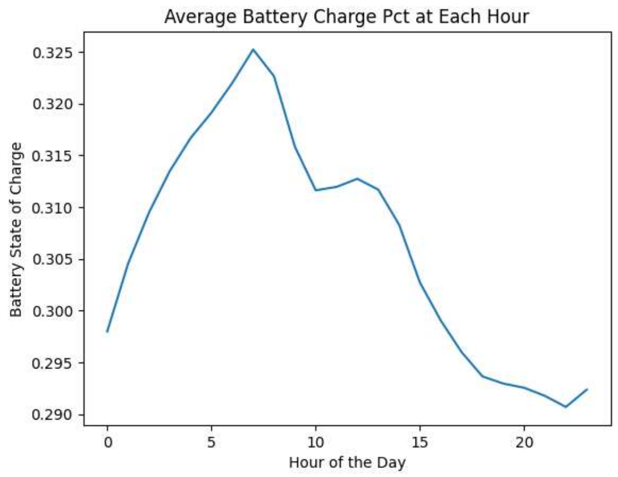
The main evaluation of whether the agent could sufficiently learn to perform optimal actions can be understood by looking at the average environment conditions:



In the morning to early afternoon, total energy generation is at or just below the demand, and then trends upwards above it. In the later afternoon, power generation decreases while price and demand increase. Therefore, a control agent would seek to lower prices in the later morning to increase demand at that time and reduce demand in the future. Additionally, it should store any excess power in the battery to prepare for the evening's deficit.

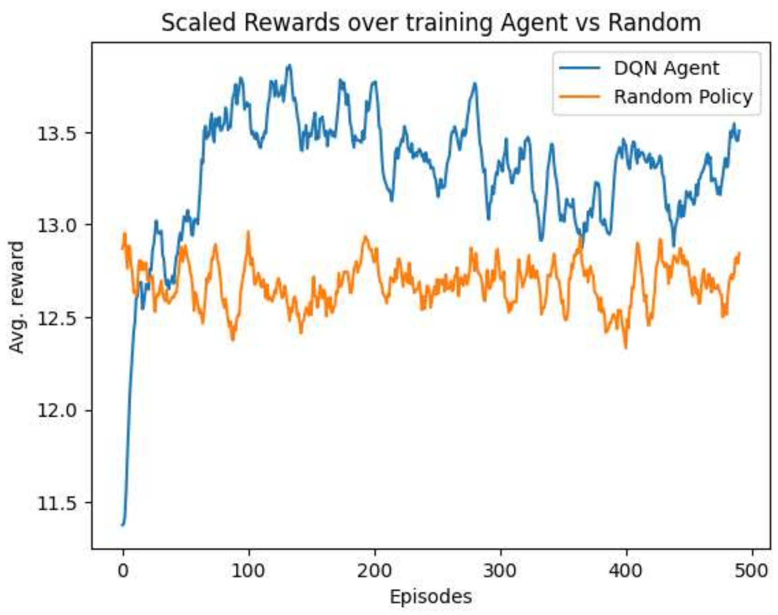
In problem 2, the agent was trained for 500 episodes, and looking at the final 150 episodes, it's clear that it was correctly learning these policies:





While the magnitudes of theses policies are not quite optimal, it is clear from the above that the agent was learning to lower the price in the first half of the day and charge the battery, and then raise the price and discharge the battery in the second half of the day.

Running agents in problem 2 also would cause my computer to crash, however, I was able to brute force through 3 iterations of 500 episodes. For these agents, I increased the replay memory so that no transitions were lost. The average of these three show that the policy being learned by the agent was effective and that the increased replay memory size helped reduce catastrophic drops in reward:



10-day smoothed average reward for average of three agents across 500 episodes vs a random policy

5 Conclusion

In this paper, I presented a microgrid environment utilizing inputs local to the Boston area. It was shown that a simple DQN reinforcement learning agent could learn an intelligent policy that managed both the pricing and charging operations of the microgrid. Microgrid control is a complex task due to the high variability in environment conditions and features. Academic and industry research is constantly being generated to evaluate more complex reinforcement learning algorithms that can handle more sophisticated and generalizable environments, particularly those that can adapt well to real-world scenarios.

There are numerous sub-experiments that could have been run just in this environment-agent combination. Due to computation and time limitations, very few hyperparameter values were varied. Values for epsilon and the learning rate, alpha could have been tested at multiple levels. Additionally, neural networks with additional hidden layers and nodes could have been evaluated, but again, this dramatically increases time and computation required in each episode. Also, replay memory, batch size, the number of forecast hours, and the size of the action space could have been varied to modify the circumstances of the environment and agent. Finally, other reinforcement learning algorithms such as SARSA, double-DQN, and policy-based algorithms could be explored in this environment. Actor-Critic models have been shown to perform especially well in these domains [2].

6 Appendix

Hyperparameter settings

	Parameter	Value
Environment		
	Grid Import Rate	1.25
	Grid Export rate	0.75
	Price Beta	-0.7
	Price Increment	0.1
	Town Size	10000
	Average Hourly Household Load	1.25 kwh
	Turbines	100
	Square Meters of Panels	100
	Battery Capacity (avg hours)	2
	Charge/Discharge Max (avg hours)	1
Agent		
	Replay Memory	3000, 12000
	Batch Size	40
	Update Frequency (Episodes)	1
	Discount Factor	1
	Learning Rate	0.1
DQN Network		
	Hidden Layers	2
	Layer Types	Dense
	Nodes/Hidden Layer	30
	Hidden Activation	relu
	Output Activation	linear
	Optimizer	adam
	Loss	mse

References

[1] N. Tomin, A. Zhukov, A. Domyshev, “Deep reinforcement learning for energy microgrids management considering flexible energy sources”, EPJ Web Conf. 217 (2019) 01016, http://dx.doi.org/10.1051/epjconf/201921701016.

[2] Taha Abdelhalim Nakabi and Pekka Toivanen, “Deep Reinforcement Learning for Energy Management in a Microgrid with Flexible Demand,” *Sustainable Energy, Grids and Networks* 25 (March 1, 2021): p. 100413, https://doi.org/10.1016/j.segan.2020.100413.

[3] Ying Ji et al., “Real-Time Energy Management of a Microgrid Using Deep Reinforcement Learning,” *Energies* 12, no. 12 (2019): p. 2291, https://doi.org/10.3390/en12122291.

[4] NSRDB. (n.d.). NSRDB: National Solar Radiation Database. Retrieved November 5, 2022, from https://nsrdb.nrel.gov/data-viewer

[5] ISO-NE: Actual Load - LCG Consulting :: EnergyOnline. (n.d.-b). http://www.energyonline.com/Data/GenericData.aspx?DataId=16