SAND2021-8232C

MLDL

Machine Learning and Deep Learning Conference 2021

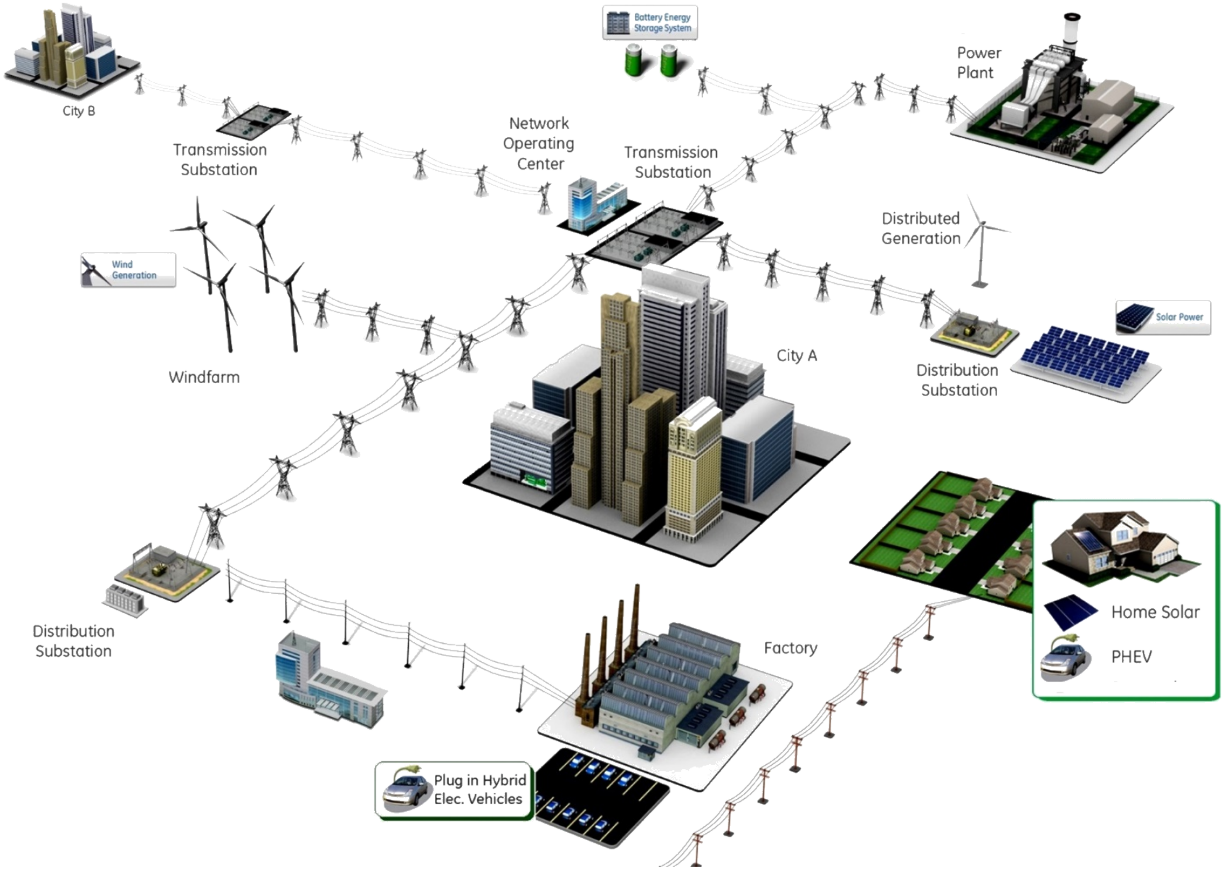
Discrete Deep Reinforcement Learning For Online Distribution Power System Cybersecurity Protection

- Tyson Bailey / 5683
- Jay Johnson / 8812
- Drew Levin / 8721

- Funding Source: LDRD

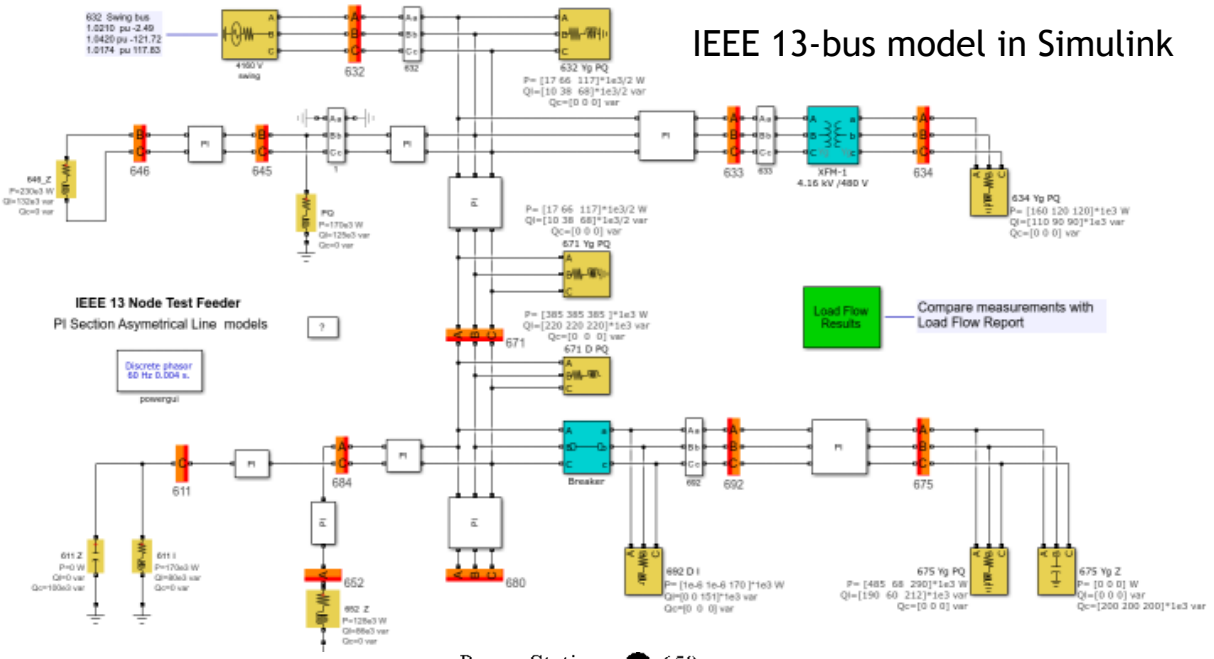
Problem

- Interconnected infrastructure exposes new and unique vulnerabilities.
- How do we respond to an adversary who is targeting the grid with informed ill intent?
- **We require a new capability to provide a dynamic response to this new threat.**

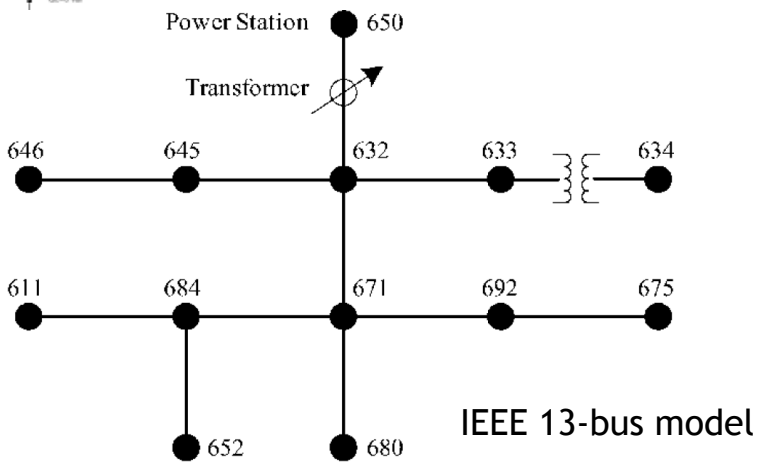


Our Scenario

- 13 Node Test Feeder – This circuit model is very small and used to test common features of distribution analysis software.
- It is characterized by being short, relatively highly loaded, with only a single voltage regulator at the substation.



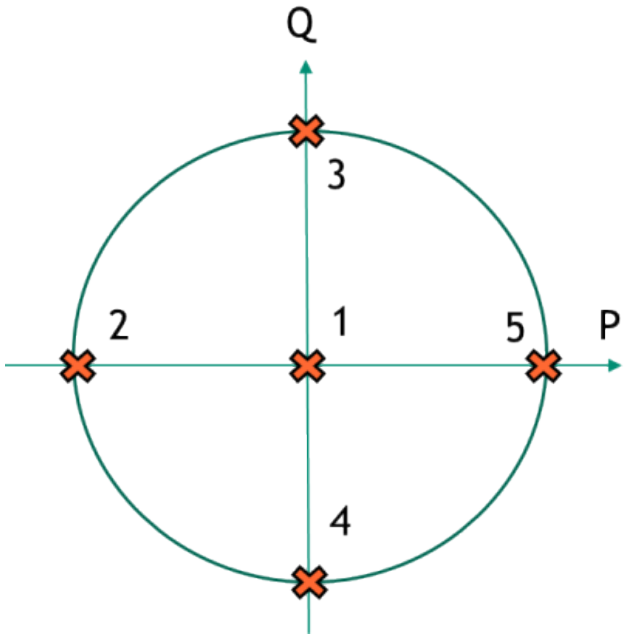
IEEE 13-bus model in Simulink



IEEE 13-bus model

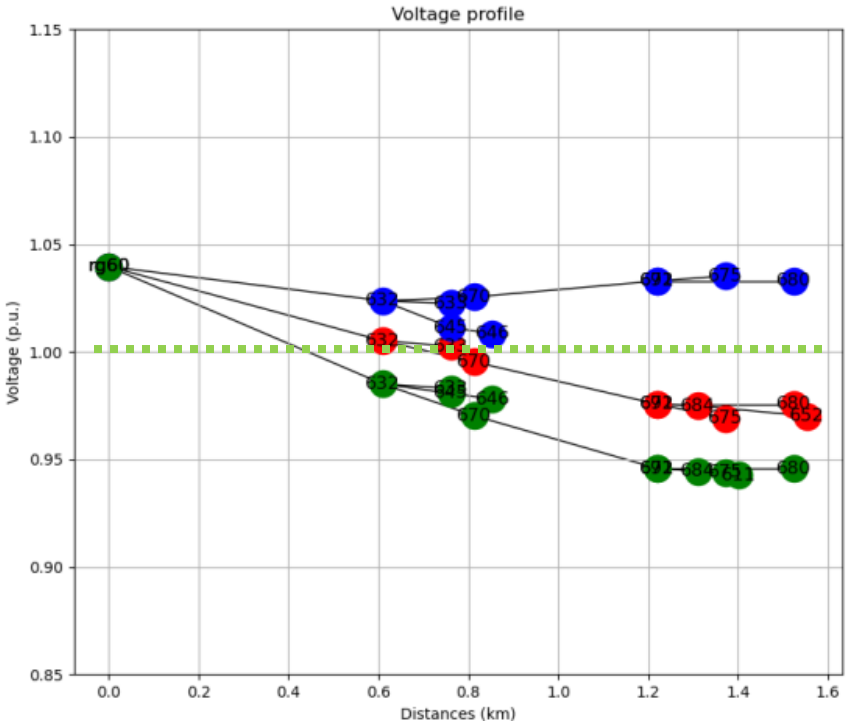
Model Controls and Objective

Each bus can be set to 1 of 5 possible states



P – Real Power
Q – Reactive Power

Goal: All buses to nominal voltage



Environment

- **Reward:** negative sum of squared errors of bus voltages compared to nominal voltage values.

$$r = -\sum_{i=0}^{n-1} (V_i - V_i^*)^2$$

where V_i and V_i^* are the voltage and nominal voltage of bus i, respectively.

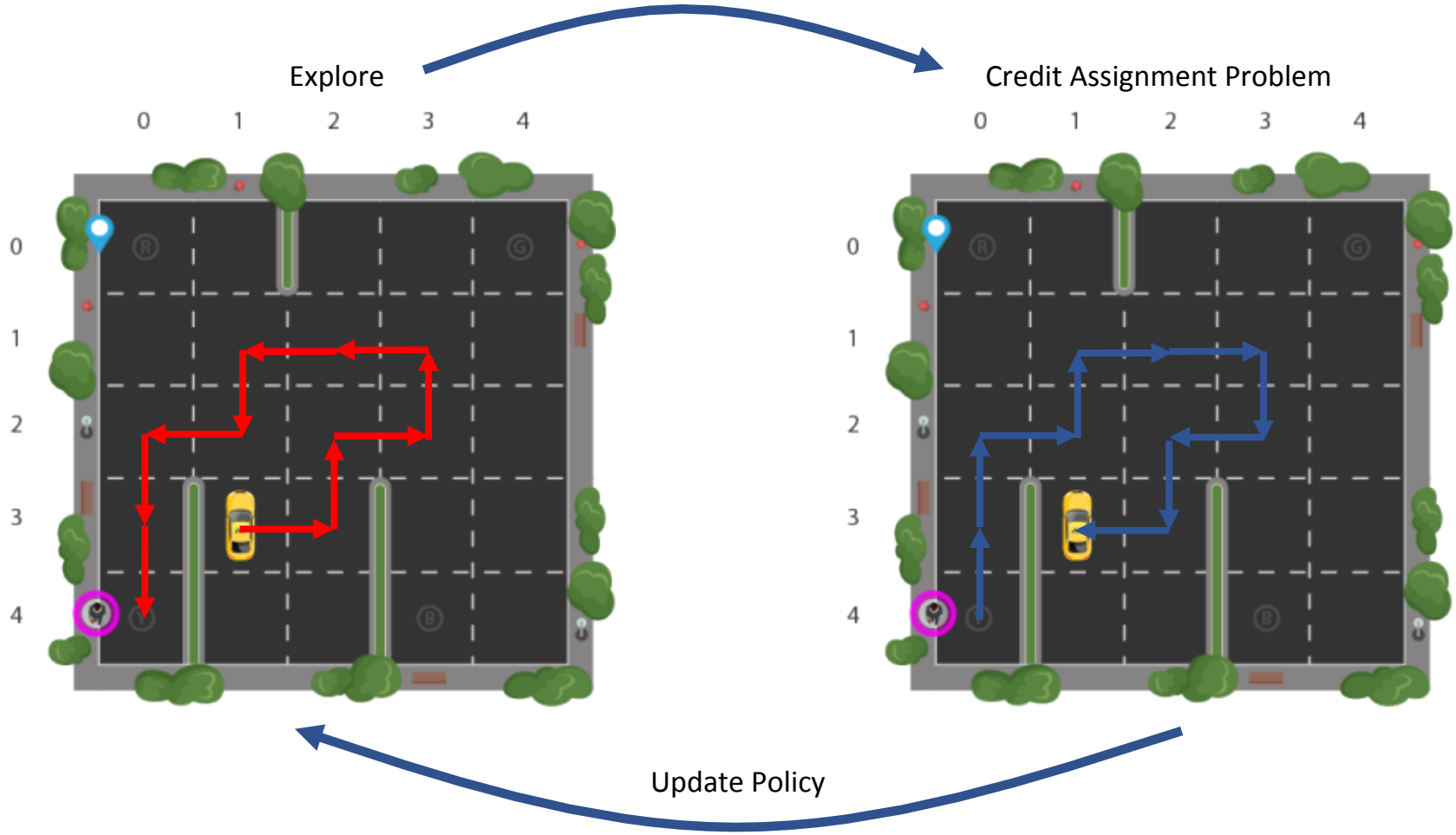
- **Horizon:** $T = 20$ steps
- **Objective:** Maximize the expected discounted reward

$$J = \mathrm{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t \right]$$

where $\gamma \in (0,1)$ is a discounting factor

- **Objective Interpretation:** Stabilize the system by bringing voltages as close to nominal as possible.

How to Learn: an RL Primer



But What About **Deep** Reinforcement Learning?

After 3 Moves - 9 million different possible positions

After 4 Moves - 288 billion different possible positions

Overall – Estimated approximately 10^50 possible positions

Supervised Learning!

Experience

Results

Policy Function
(Deep Neural Net)

$$\pi_\Theta(S_n) \rightarrow A_n$$

Policy Parameterization
(Neural Net Edge Weights)

Deep Q-Learning Rapid Explanation

A Deep Q-Network is a simple DRL algorithm for discrete environments.

`Quality' Function (Deep Neural Net)

Update scaled by the learning rate

Future rewards are discounted

All compared vs. the current estimate

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Updating from current value

We consider the immediate reward and the estimated future rewards (recursively)

DQN - Details

Values we experimented with

Parameters	Final Values	Range
Gamma	.9	0-0.9
Learning Rate	1e-5	1e-5 – 1e-2
Replay Memory	50000	10k – 200k
Target Update Delay	50	10 -500
Max Epsilon	.9	0.8-1.0
Min Epsilon	.1	0-0.2
Epsilon Decay	10000	2k – 100k

Values that achieved the best results

Experimentation – Details

0. Set Random Seeds

1. Selected Hyperparameters

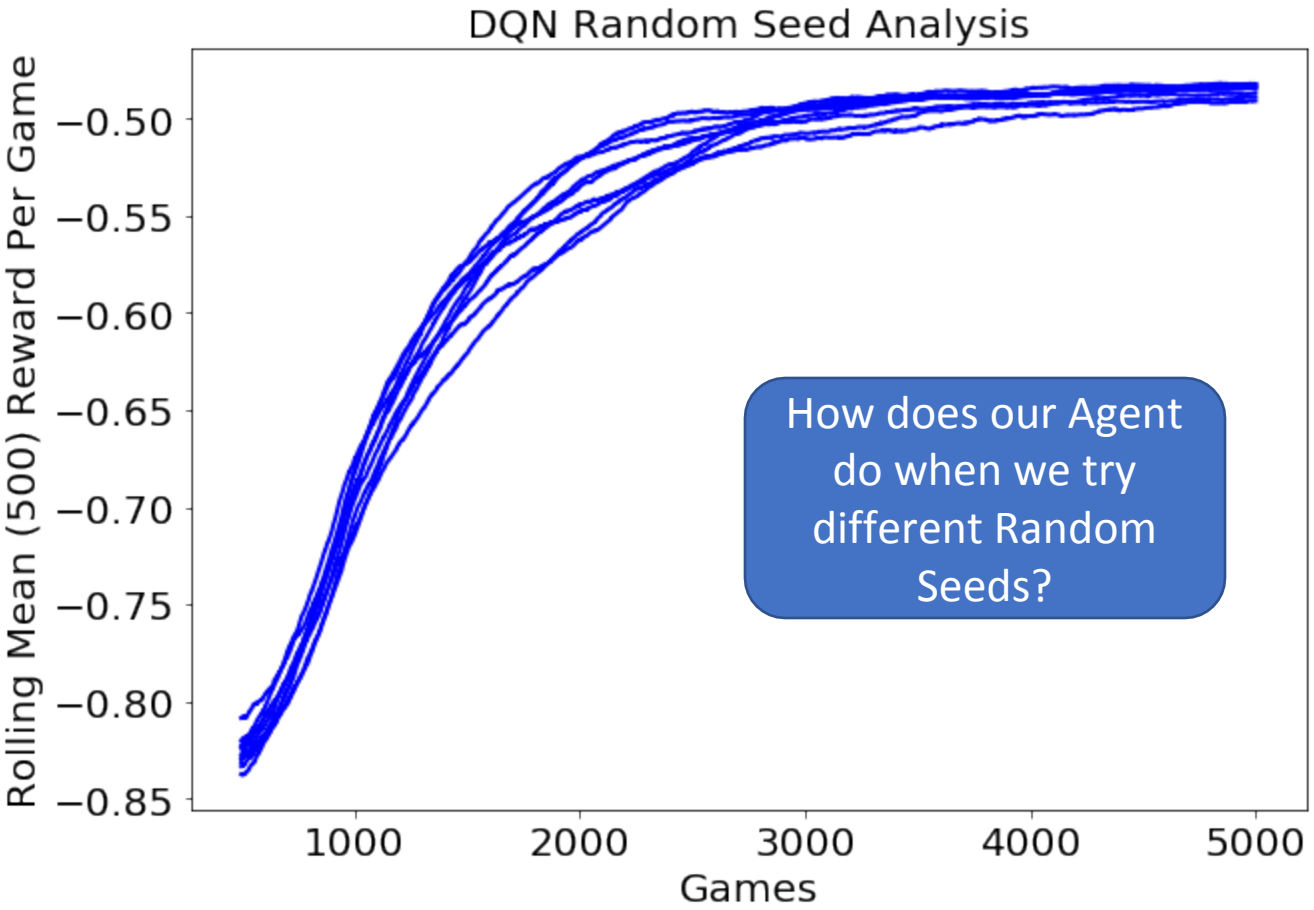
2. Started Game with a default start state

3. Ran game taking 20 steps

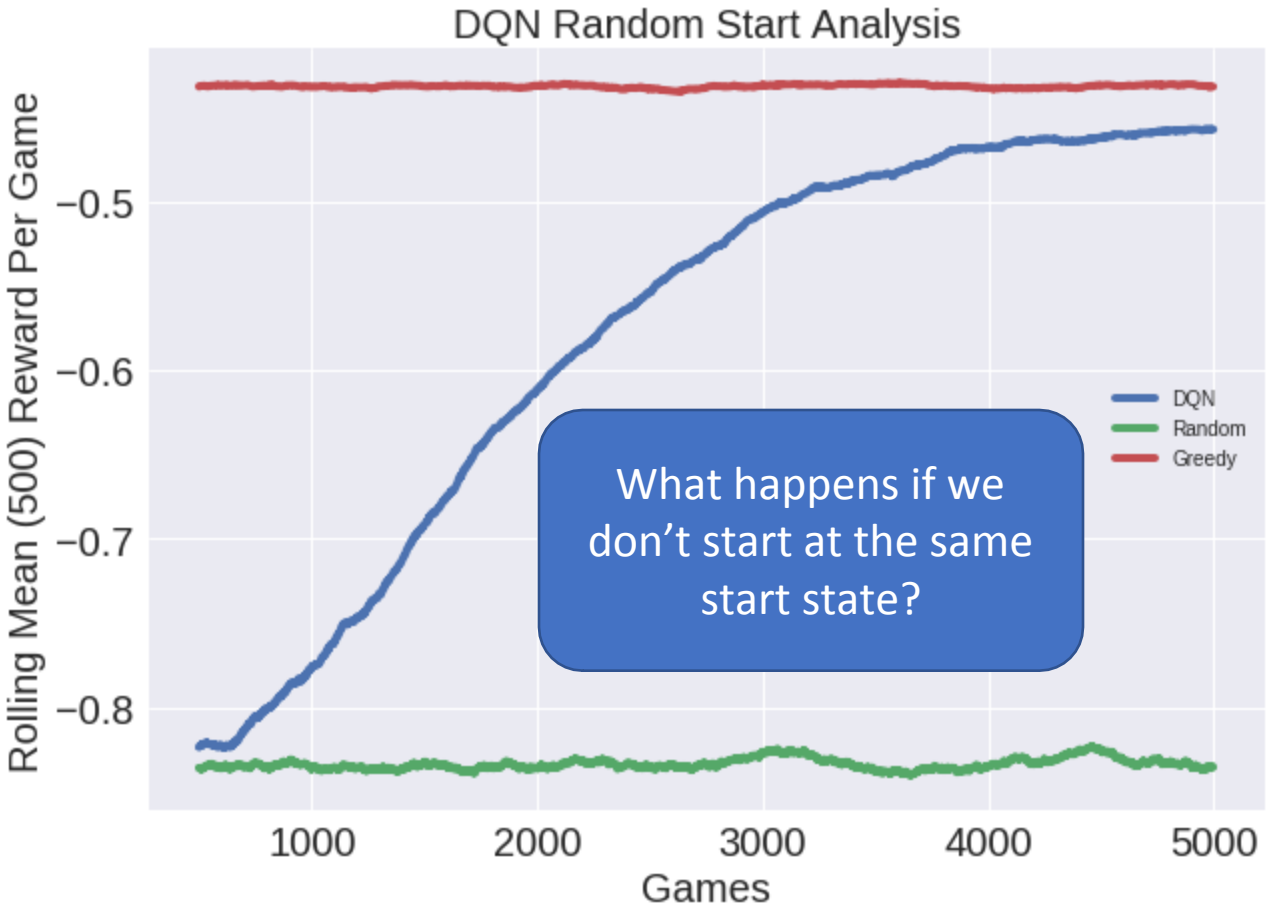
4. Compared to a Greedy Strategy

5. Repeat at Step 1 if drastically worse

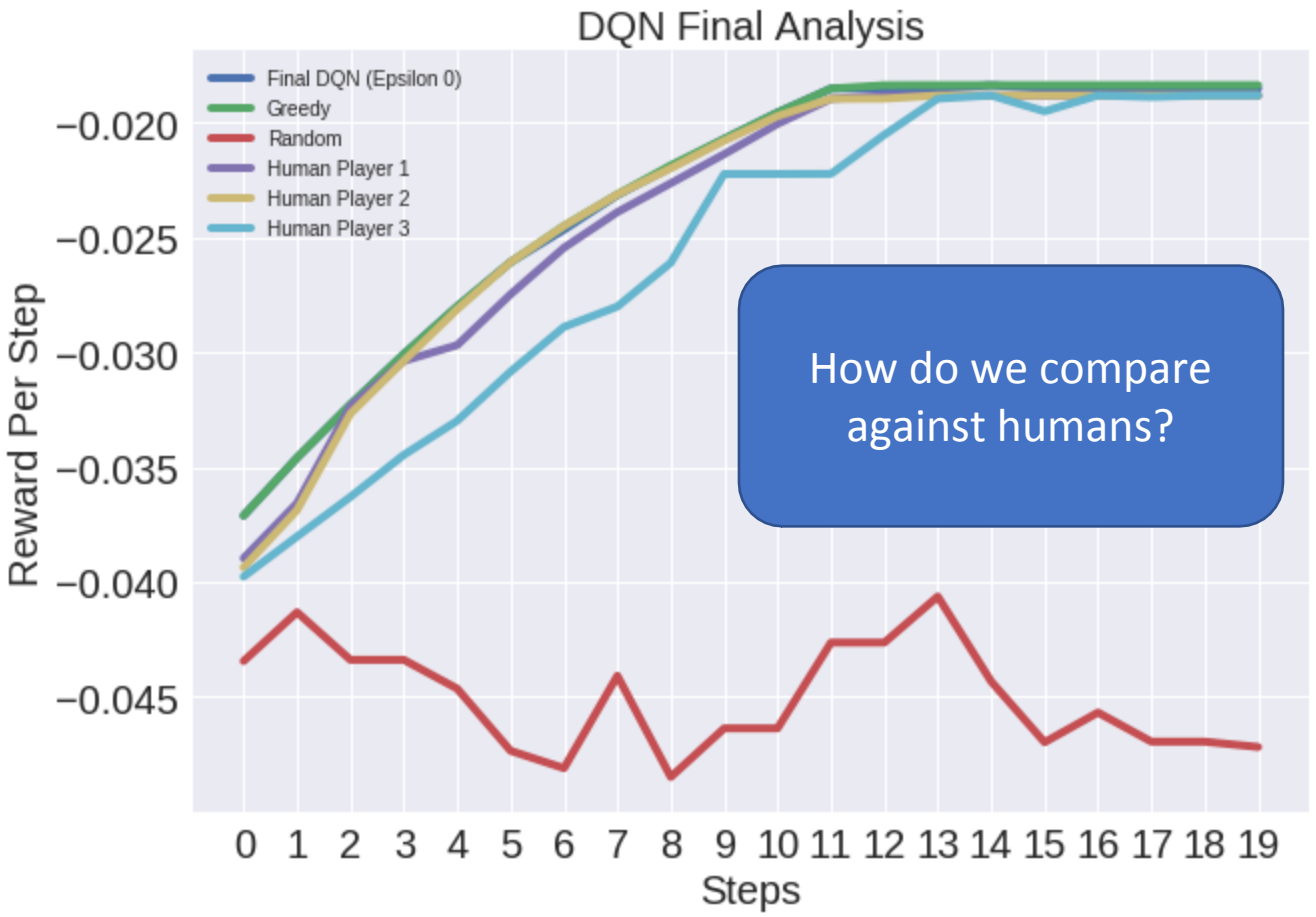
Experimentation – Details



Experimentation – Details



Final Results – Details



Final Results – Timing

Timing	Greedy	DQN
Real	0.811s - 0.865s	1.397s – 2.433s
User	0.707s - 0.689s	1.088s – 1.161s
sys	0.077s - 0.105s	0.239s – 0.375s

Future Work

- Continuous Control
- Multi-player (Adversary/Defender)
- Increasing time between actions
- Partial and/or Delayed Observability
- Variable DER Active Power Capacities
- Experimenting with reward shaping
- Increasing the number of DER and/or the size of the OpenDSS power system model