INDO-GERMAN CENTRE FOR SUSTAINABILITY (IGCS)

GRANTEE EXPERIENCE REPORT

Indo-German Centre for Sustainabilty (IGCS)
IGCS Research Exchange, Grant Period 2023

research brief

Nico Dabelstein

Research Brief from IGCS Grant Period 2023

IGCS Grantee
Nico Dabelstein

Home Institute
Technical University Berlin, EVUR

Host Institute | Host Supervisor
Dep. of Electrical Engineering,
Indian Institute of Technology Madras, Chennai, India
Supervised by: Prof. Dr. Krishna Vasudevan

Research Topic
Sustainable Urban Development Microgrid Control using Deep Reinforcement Learning

Starting/End date of the student exchange period
15.05.2023 – 15.08.2023

About the IGCS Grants

IGCS awards scholarships to students and researchers from India and Germany with excellent academic records, very good English, and intercultural communication skills. The scholarship consists of a mobility grant and an accommodation grant according to DAAD funding rates, as a rule. Learn more about the funding opportunities at IGCS here.

Research Brief from IGCS Research Exchange, Funding Period 2023

Professor Krishna Vasudevan leads a research group at IIT Madras which is developing a microgrid on campus. There are different departments and research projects involved focusing on various aspects of the microgrid. The focus of my research is on the development of a Deep Reinforcement Learning (DRL) agent for scheduling and optimizing energy distribution within the microgrid. DRL is a machine learning approach that uses trial and error along with rewards to determine policies to achieve predetermined goals. The research objectives include:

- Creating a DRL algorithm for efficient microgrid control.
- Enhancing microgrid effectiveness by optimally distributing electricity among its components.
- Improving the microgrid's environmental and economic performance.

The methodology of the project started with the gathering of necessary data at IITM to create a representative environment of the microgrid on campus. After the data collection, the components of the microgrid were modeled in Python, such as the photovoltaic system, the steam turbine, a battery, and the grid connection. For the environment, Python libraries like stablebaselines3 and pandas were used. The DRL agent interacts with the environment according to its action space, a set of options that includes all possible actions the agent can choose from, and the observation space a set of observations the agent can make from the environment. Additionally, a complex reward function was created that considers factors like costs, CO2 emissions, and battery state of charge.

Once the environment, the action and observation space, and the reward function were set the initial training of the DRL agent was started. The training revealed that the performance was suboptimal and different means of improvement were discussed like normalizing the observation space and hyperparameter tuning. Furthermore, it was determined that the environment was too simplistic. To overcome this the decision was made to implement a Pandapower Python model. Pandapower allows the accurate modeling of components of a microgrid and Power Flow and Optimal Power Flow (OPF) calculations. OPF calculations are time-consuming and DRL agent training requires many episodes which would make OPF calculations after every time step too time-consuming. Hence, an artificial neural network (ANN) was introduced to approximate OPF, allowing fast approximations of the optimal power flow which can then be included in the training of the DRL agent.

The project is part of a PhD and is still ongoing. Future steps will include further refining of the observation and action space, integrating the Pandapower environment in the old DRL framework, and expanding the ANN training to include diverse scenarios. The final goal is a robust and precise control algorithm that has an optimal or near-optimal solution for the control of the microgrid to manage fluctuating renewable energy and demand.

1. Introduction

A research group around Professor Krishna Vasudevan is working on building and developing a microgrid at the IIT Madras campus. There are different departments involved in the development of the new microgrid focusing on various aspects and components of the system. The goal of my research project is to find a suitable control algorithm for scheduling and optimizing the energy dispatch within the microgrid. Deep reinforcement learning is used to achieve this. Deep Reinforcement Learning (DRL) is a machine learning algorithm that uses trial and error and rewards to produce a suitable policy that achieves the goals that have been defined. Figure 1 shows the general layout of how a DRL agent works. A Reinforcement Learning agent interacts with an environment and makes a sequence of actions. The agent can choose actions that are defined within the action space.

The action space is the set of all actions the agent can take.

Depending on the chosen action and the resulting observations the agent receives rewards and penalties. Each action affects the environment and thus creates a new observation and state for the next time step. A Deep Neural Net (DNN) is used for the approximation of the policy and is adjusted at every time step during training. The agent selects actions based on its current policy. Over time, the agent learns the best policy for its actions through trial and error. In the same way, the action space needs to be defined, the observation space must also be defined. The observation space is what the agent can observe and is also used to update the agent's policy function. The measurement for a good policy is the cumulative reward that is achieved at the end of every training episode. The reward is the signal the agent receives from the environment in response to its actions.

Hence the goal of the agent is to maximize the cumulative reward over time and improve the policy with each training episode. The design of the action, observation, and reward space is crucial to the DRL agent's performance, and finding an optimal solution can be time-consuming.

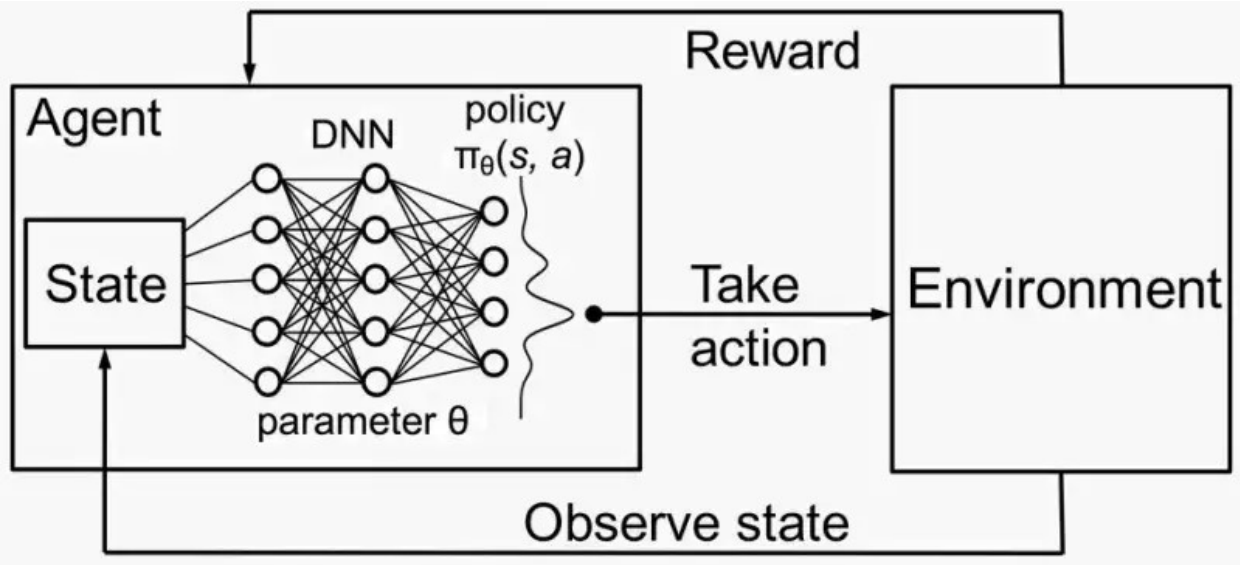

Figure 1: Working principle of a DRL algorithm. [1]

2. Research objectives

For the development of the DRL agent for the microgrid the following research objectives were defined:

Development of a DRL Algorithm for Microgrid Control: The primary goal is to develop a Deep Reinforcement Learning algorithm specifically tailored to manage and optimize power distribution within a microgrid.

Maximization of Microgrid Effectiveness: The DRL algorithm will be trained to distribute electricity efficiently among various components of the microgrid such as waste-to-energy systems, solar PV systems, and flow battery systems to manage variability in power generation.

Optimization of Environmental and Economic Performance: To train the DRL agent to optimize the microgrid's operations not only for energy efficiency and stability but also to minimize operational costs and CO2 emissions.

3. Methodology and project development

The beginning of the project was characterized by obtaining data about said microgrid. The environment that the agent will interact with is designed to reflect the real microgrid as much as possible. Therefore, the collected data was used to scale the components within the environment. The general layout of the microgrid can be seen in Figure 2. The microgrid consists of a photovoltaic system, a steam turbine, different loads, a battery, and a grid connection. The model is created in Python using several different libraries like stablebaselines3, pandas, NumPy, and more.

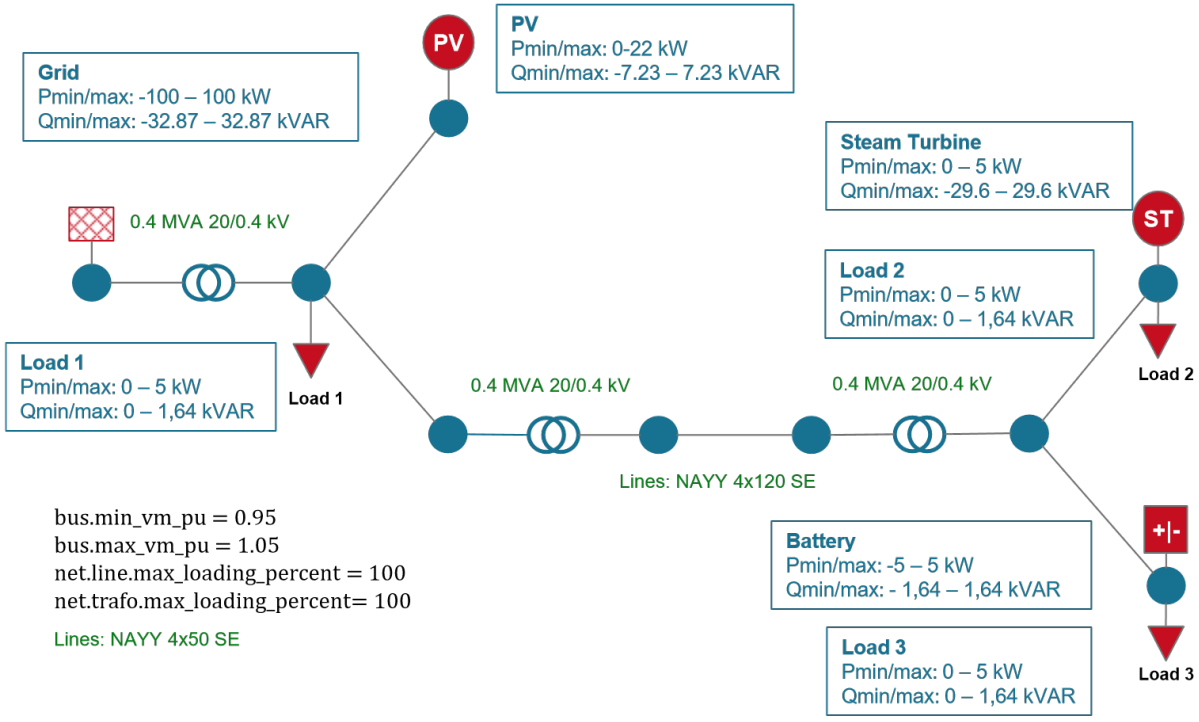

Figure 2: Microgrid Structure

Additionally, to the components of the microgrid itself, the individual elements of the environment that the agent depends on must be configured, like the action and observation space and the reward function. The Action space is set to two actions which are the steam turbine action and a battery action. The steam turbine action is defined as a range between 0 and 1 and represents the percentage of the output power of the steam turbine. The battery action is set to a range between -1 and 1 which represents charging and discharging. The observation space includes the current power output of the photovoltaic system, the load demand, the current state of charge (SOC) of the battery, the last power setting of the battery, the grid, and the steam turbine, the price for buying and selling electricity from the national grid and the CO2 emissions resulting from the last actions. And lastly, the reward function is defined. The reward is calculated based on several factors like the cost and CO2 emissions, the battery actions and steam turbine actions, and the amount of grid interaction as well as the state of charge of the battery, resulting in the following reward function:

$$R = -w1 \times \text{cost} - w2 \times \text{CO2} - w3 \times |Abat,t - Abat,t-1| + w4 \times AST - w5 \times Egrid2\_buy - w6 \times Egrid2\_sell - SOC_{penalties}$$

Where $w_1, w_2, \ldots, w_6$ are weights representing the importance of each factor that are adjusted to achieve the goals according to their priority. A_{ST} and A_{bat} are the steam turbine action and the battery action. E stands for the energy sold or bought from the grid. SOC penalties are imposed when the battery's SOC goes outside its operational range. Figure 3 shows how the first iteration of the model worked.

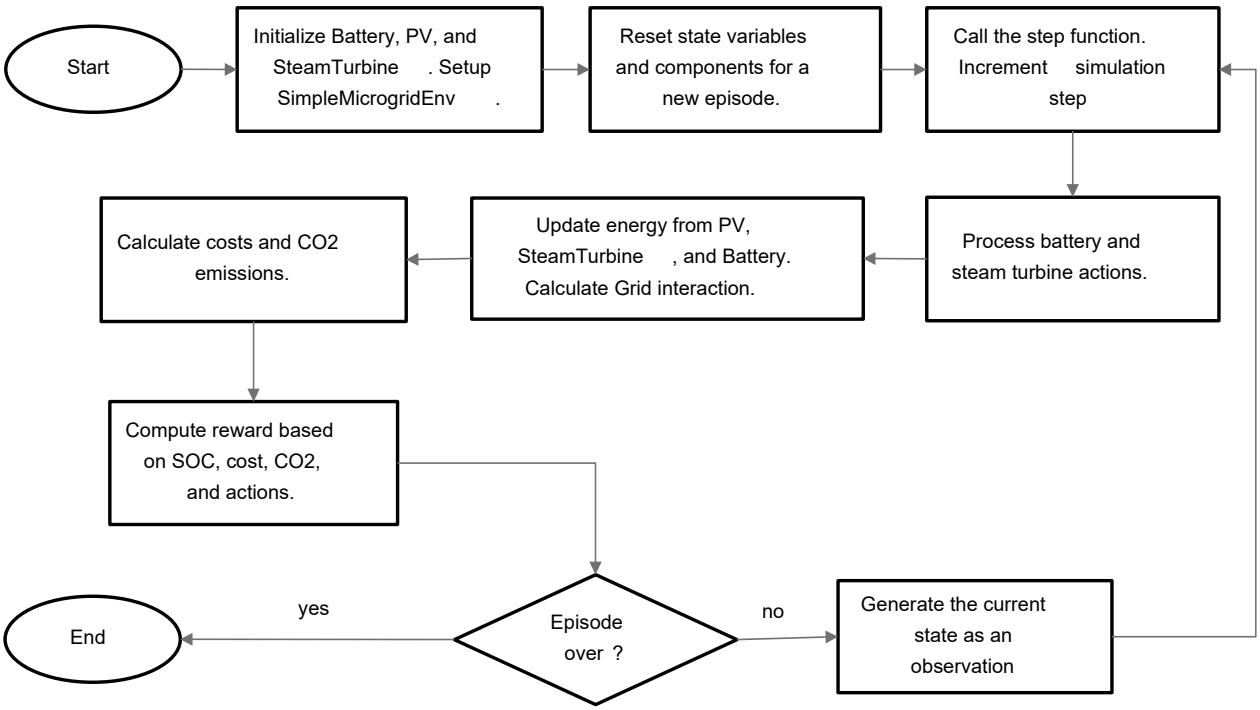

Figure 3: Model Structure

4. Initial results and model improvement

After the environment was designed several rounds of training were conducted resulting in an agent that behaves as seen in Figure 4. The agent successfully controlled the environment but on closer investigation of the results it could be determined that the results were far from an optimal solution to the problem.

Even after a high amount of training the results are not satisfactory since the grid interaction is still high even in times when there is no need. The agent is discharging the battery and running the steam turbine at the same time and selling the excess energy it is generating to the national grid. This results in higher costs and CO2 emissions than necessary.

To tackle these problems several ways were discussed. A first improvement could be the normalization of the observation space which could allow better judgment for the agent since only values between 0 and 1 would exist for it to process. Second would be the hyperparameter tuning, those values influence how the agent learns, for example, how greedy it is to get immediate rewards or if it is more focused on rewards later in the training episode. And lastly further improvement of the reward function and more training.

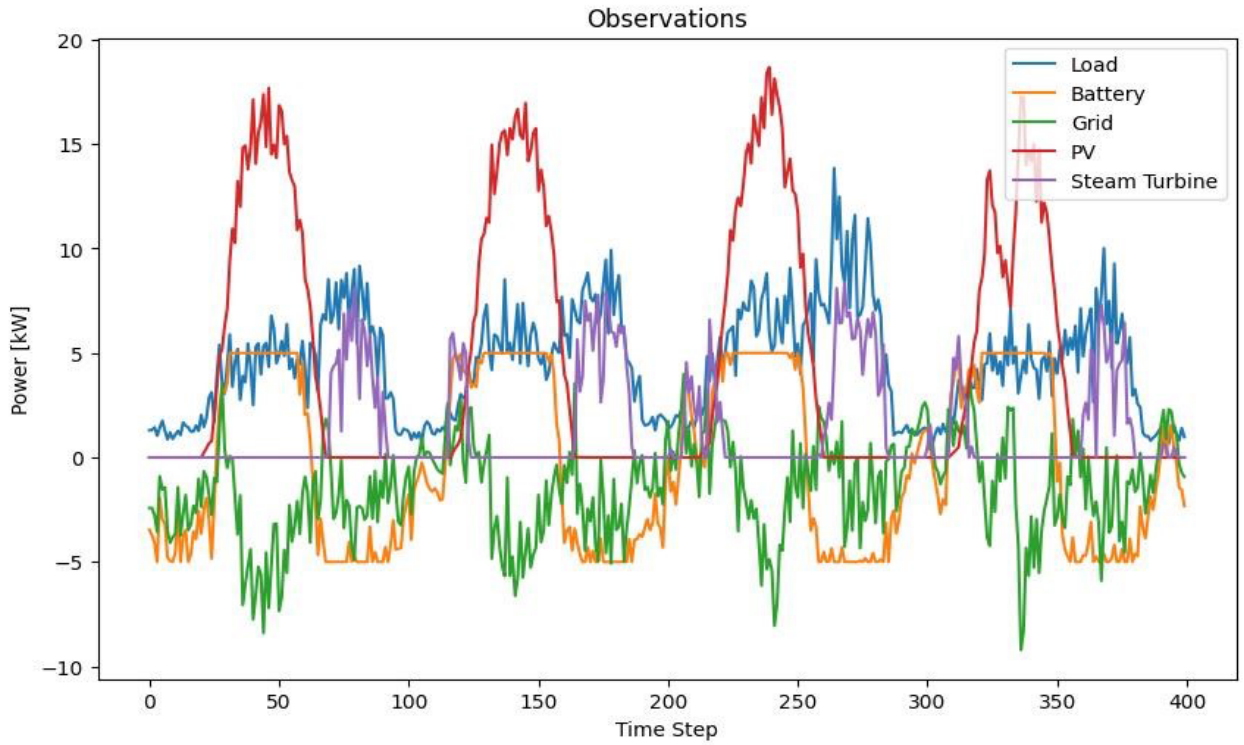

Figure 4: Results of the initial model

The second observation that was made is that the initial environment was relatively simple and didn't include all the restrictions and behaviors real microgrid components have. Since the environment did not necessarily follow the actual physical limitations of the microgrid a Pandapower Network with Power flow and optimal power flow calculation was discussed. Pandapower is a Python library that allows the design of accurate power systems and performs power flow calculations after the parameters of components are determined.

Additionally, Pandapower provides the possibility to conduct optimal power flow calculations to determine the optimal component parameters. Furthermore, Pandapower allows the creation of components like transformers, batteries, and generators which allows a more accurate representation of the microgrid with power flow calculations after the agent chooses actions. A problem with OPF calculations is the long computing time. Long computing times are a big disadvantage for this project due to the vast amount of training steps that will be needed for the DRL agent. To avoid the long computing time after every timestep an artificial neural network (ANN) is used to approximate the OPF after every action that the agent chose. The results can be used to determine how close to the optimal solution the agent got.

To be able to use the ANN for OPF approximation the ANN first has to be trained. For training data generation, a different program was coded that creates a training set and a test set with several ten thousand data points of OPF calculations. These training sets will then be used to train the ANN. Here it is important to include OPF calculations for a wide variety of scenarios and timesteps to make sure that the ANN can provide reliable results even for uncommon scenarios.

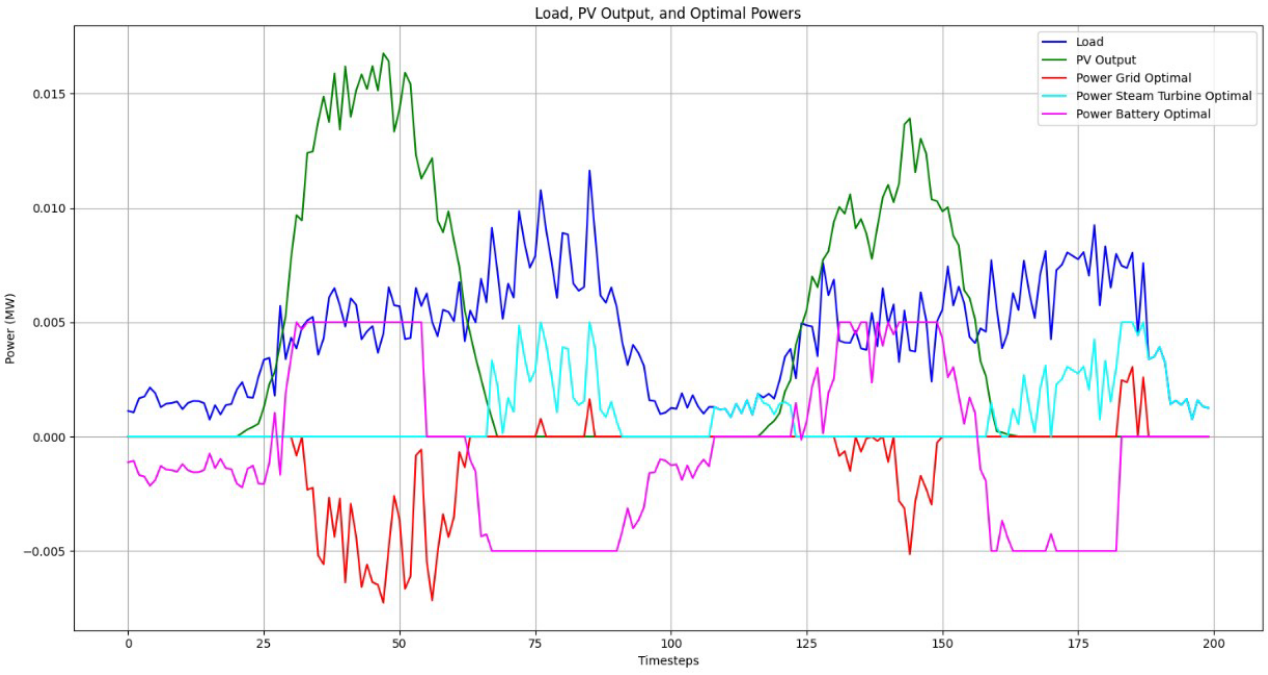

Figure 5: Optimal Power Flow results

Figure 5 shows the results of an optimal power flow calculation for 200 timesteps. As can be seen when comparing Figure 4 and Figure 5 the grid interaction is reduced to a minimum which shows the possible improvement that can be achieved once the agent achieves optimal or near optimal solutions. The ANN can predict the OPF according to the current observations and the OPF predictions allow a better judgment of the agent's actions by including the OPF predictions in the reward calculation.

The approach of combining DRL and ANN could have several advantages resulting in a robust and precise control algorithm. The DRL agent adapts to fluctuating demands and renewable energy generation. Balances immediate and future rewards, making it ideal for battery management while the ANN for OPF allows Fast approximation to traditional optimization methods, ensuring quick optimal power flow solutions.

5. Discussion of the results and outlook

The new and improved model structure can be seen in Figure 6. The new structure now includes the OPF approximation with the ANN and shows how the improved training of the DRL agent will work. It also includes a rule that will terminate the training episode in case of violations of network constraints which should lead to a state where the agent tries to avoid those scenarios since longer episodes can potentially secure higher rewards.

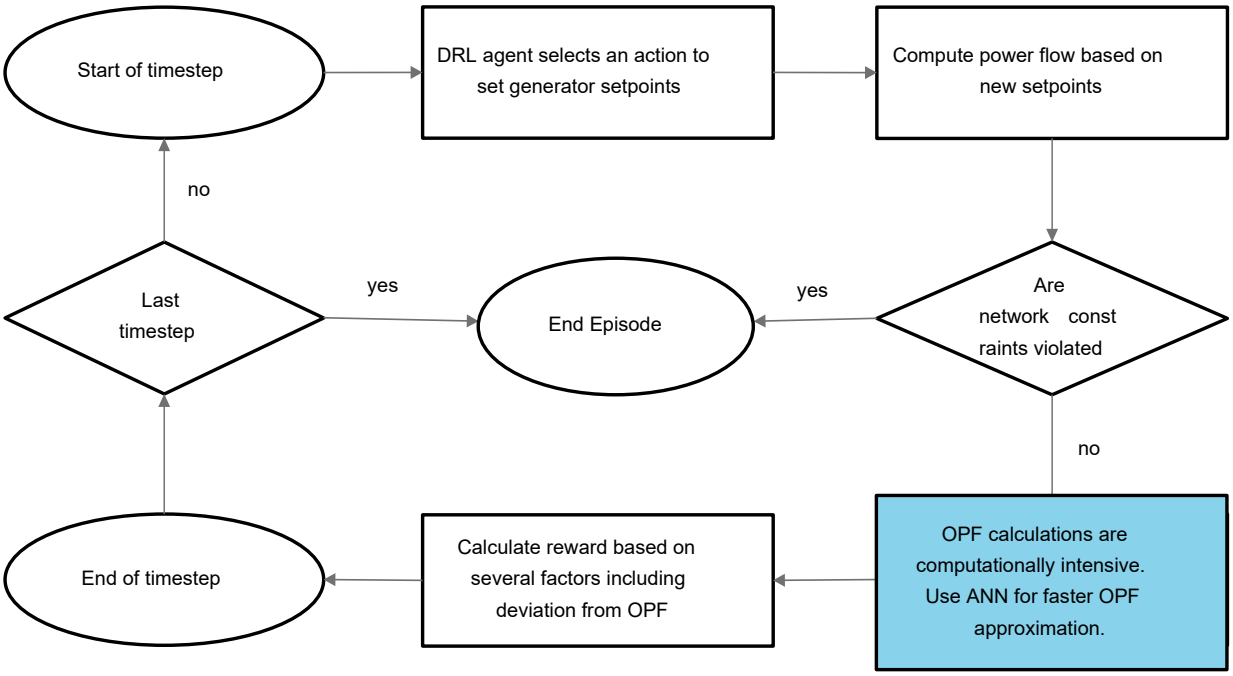

Figure 6: Logic Flow of the new model structure

The project is still ongoing, and the next steps will include the normalization of the observation and action space. Further improvement and final integration of the new Pandapower environment into the old DRL environment to combine the ANN and DRL environments. To achieve a robust ANN further training is needed that includes all kinds of scenarios like blackouts, different weather conditions, and more. The OPF results then have to be included in the reward function.

This research project is part of my ongoing PhD. So far, no publications have resulted from this work.

I want to thank Prof. Krishna Vasudevan Prof. Satyanarayanan Seshadri and their team who helped immensely with their feedback and the provision of the needed data for the project.

References

[1] A. B. P. M. Y. N. V. K. I. &. S. O. Gozhyj, Hybrid Power Plant Control System Based on Machine Learning Methods. In N. Shakhovska & M. O. Medykovskyy (Eds.), Advances in Intelligent Systems and Computing V (pp. 251-262), Springer International Publishing, 2021.