pymgrid: An Open-Source Python Microgrid Simulator for Applied Artificial Intelligence Research

Gonzague Henri*
Total EP R&T
Houston, TX
gonzague.henri@total.com

Tanguy Levent
Ecole Polytechnique, CNRS, IP Paris
Palaiseau, France

Avishai Halev
University of California, Davis & Total EP R&T
Davis, CA, USA

Reda Alami
Total S.A.
Palaiseau, France

Philippe Cordier
Total S.A.
Palaiseau, France

Abstract

Microgrids – self-contained electrical grids that are capable of disconnecting from the main grid – hold potential in both tackling climate change mitigation via reducing CO_2 emissions and adaptation by increasing infrastructure resiliency. Due to their distributed nature, microgrids are often idiosyncratic; as a result, control of these systems is nontrivial. While microgrid simulators exist, many are limited in scope and in the variety of microgrids they can simulate. We propose ***pymgrid***, an open-source Python package to generate and simulate a large number of microgrids, and the first open-source tool that can generate more than 600 different microgrids. ***pymgrid*** abstracts most of the domain expertise, allowing users to focus on control algorithms. In particular, ***pymgrid*** is built to be a reinforcement learning (RL) platform, and includes the ability to model microgrids as Markov decision processes. ***pymgrid*** also introduces two pre-computed list of microgrids, intended to allow for research reproducibility in the microgrid setting.

1 Introduction

Microgrids are defined as "a cluster of loads, distributed generation units and energy storage systems operated in coordination to reliably supply electricity, connected to the host power system at the distribution level at a single point of connection, "the point of common coupling" (PCC)" (Figure 1) [18]. First introduced in 2001, microgrids can also be completely autonomous and disconnected from the grid (off-grid) [15].

Microgrids are one of the few technologies that can contribute to both climate change mitigation - by decreasing greenhouse gas emissions – and adaptation, by increasing infrastructure resiliency [22] to extreme weather events. Further research is still necessary to fully integrate renewables and reduce costs in order to make widespread microgrid adoption feasible. Today, one billion people do not have access to electricity; this technology can be used to bring clean energy to communities that are not yet connected to the grid. The importance of bringing clean energy to these communities extends beyond the impact of climate change: indoor pollution, notably the use of dirty fuel for cooking, is a major public health challenge in the developing world [10].

Due to their distributed nature, microgrids are heterogeneous and complex systems, potentially equipped with a wide range of generators and employed in a myriad of applications. In addition, as load and renewable generation are stochastic, microgrids require advanced metering and adaptative

*Corresponding author.

Tackling Climate Change with Machine Learning workshop at NeurIPS 2020

control to maximize their potential. Current technical limitations include the amount of solar that can be integrated while the microgrid operates in islanded mode – the controller needs to maintain stability and power quality – as well as the standardization of such controllers, necessary in order to bring down cost and accelerate their deployment [11]. Furthermore, as the grid becomes increasingly digital, ever increasing data processing capabilities are required [4]. A motivation for this package is to develop tools that can best integrate with the grid of the future.

Microgrid control can be categorized in three main levels. In primary control, voltage and frequency are controlled in a to sub-second time scale. At the secondary level, control focuses on steady state energy control to correct voltage and frequency deviation. Finally, tertiary control – the focus in ***pymgrid*** – concerns itself with the long term dispatch of the various generators for optimizing the operational cost of the microgrid.

In our review of the literature surrounding tertiary control, we observed two main limitations: first, a lack of open-source microgrid models, and second, a lack of a standard dataset to benchmark algorithms across research groups (along with standard performance measurement). Together, these lead to a lack of reproductibility and a difficulty in validating published algorithms.

In this paper, we introduce ***pymgrid***, an open-source python package that serves as a microgrid virtual environment.

Through ***pymgrid***, we propose two list of pre-compute microgrids, `pymgrid10` and `pymgrid25`. Our intention is for them to be used as benchmark scenarios for algorithm development, allowing for more robust research reproductibility.

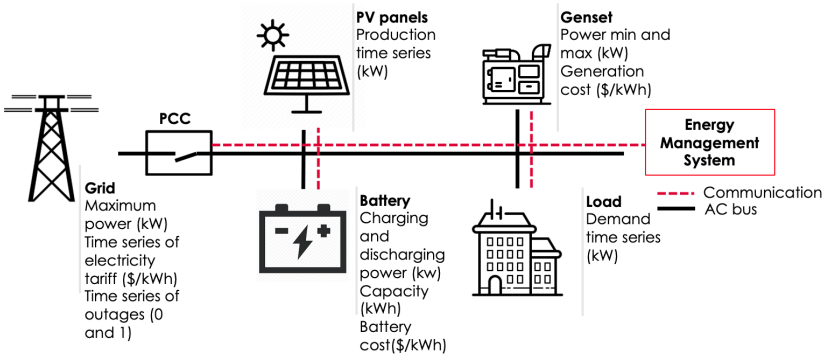

Figure 1: Overview of a microgrid

2 Prior Work

Open source Python power systems simulators exist; however, they are often limited in scope [23, 9]. Considerations of microgrids in the literature focus on large-scale power systems [19, 7]. An open-source simulator in the OpenAI gym environment, representing a microgrid for RL, exists, but targets primary control applications [6, 8]. Other models are available on GitHub but either do not simulate tertiary control, are difficult to scale to multiple microgrids or do not allow for straightforward RL integration [1, 2, 3]. To the best of the authors knowledge, there does not exist an open source simulator for a large number of microgrids focusing on tertiary control as of September 2020.

As discussed in [20], machine learning (ML) algorithms hold promise in power systems and grid related topics. RL and control are listed as relevant for enabling low-carbon electricity. Historically, the field has leveraged ML for forecasting, anomaly detection or uncertainty quantification [5].

However, with recent advances in RL and algorithms beating the world champion of Go, an increasing number of researchers are interested in applying RL to grid-related control applications [21]. In [12, 25, 24, 17], the authors propose reviews of reinforcement learning applications to power systems and the grid. In 2020, the Learning to Run Power Networks Challenge will take place, challening the community to build a Reinforcement Learning agent to manage the real-time operations of a power grid [14].

3 Pymgrid

pymgrid consists of three main components: a data folder containing load and PV production time series that are used to 'seed' microgrids, a microgrid generator class named ***MicrogridGenerator***, and a microgrid simulator class called ***Microgrid***.

3.1 Data Collection

In order to easily generate microgrids, ***pymgrid*** ships with load and PV production datasets. The load data comes from DOE OpenEI [2], and is based on the TMY3 weather data; the PV data is also based on TMY3 and is made available by DOE/NREL/ALLIANCE [3]. These datasets contain a year long timeseries with a one hour time-step for a total of 8760 points. Included in the datasets are load and PV files from five cities, each within a different climate zone in the US.

3.2 Microgrid

This class contains a full implementation of one microgrid; it contains the time series data and the specific sizing for one microgrid. ***Microgrid*** implements three families of functions: the control loop, two benchmark algorithms, and utility functions.

A few functions need to be used in order to interact with a ***Microgrid*** object. The function **`run()`** is used to move forward one time step, it takes as argument a control dictionary and returns the updated state of the microgrid. The control dictionary centralizes all the power commands that need to be passed to the microgrid in order to operate each generator at each time-step. Once the microgrid reaches the final time-step of the data, its $done$ argument will pass to $True$. The **`reset()`** function can be used to reset the microgrid at its initial state, emptying the tracking data structure and resetting the time-step. Once a control action is applied to the microgrid, it will go through checking functions to make sure the commands respect the microgrid constraints.

For reinforcement learning benchmarks and more generally for machine learning, another extra function is useful, **`train_test_split()`** allows the user to split the dataset in two, a training and a testing set. The user can also use **`reset()`** to go from the training set to the testing set, using the argument $testing = True$ in the reset function. An example is provided in Listing.

3.3 MicrogridGenerator class

MicrogridGenerator contains functionality to generate a list of microgrids. For each requested microgrid, the process proceeds as follows. First, the maximum power of the load is generated randomly. A load file is then randomly selected and scaled to the previously generated value. The next step is to automatically and randomly select an architecture for the microgrids. PV and batteries are always present – this might evolve in the future as we add more components – and we randomly choose if we will have a diesel generator (genset), a grid, no grid or a weak grid (a grid-connected system with frequent outages). In the case of a weak grid, we also implement a back-up genset; if there is either a grid or a weak grid, an electricity tariff is randomly selected. The electricity tariffs are generated by ***MicrogridGenerator***, and are based on commercial tariffs in California and France.

Once the architecture is selected, the microgrids need to be sized. First, a PV penetration is calculated (as defined by NREL as load maximum power / PV maximum power, in [13]) and this value is used to randomly scale the selected PV profile. Generated grid sizes are guaranteed to be larger than the maximum load, and the genset provides enough power to fulfill the peak load. Finally, batteries are capable of delivering power for the equivalent of three to five hours of mean load.

Once the different components are selected and sized, ***MicrogridGenerator*** creates a ***Microgrid***. This process is repeated to generate a number of user-requested microgrids.

Overall, with five load files, five PV files, two tariffs, three types of grid, and the binary genset choice, the model can generate more than 600 different microgrid – before even considering the number of possible different PV penetration levels.

4 Benchmarks and Discussion

In addition to the aforementioned capabilities, we proposing two standard microgrid lists: `pymgrid10` and `pymgrid25`, containing 10 and 25 microgrids, respectively. `pymgrid10` has

[2] `https://openei.org`

[3] `https://rredc.nrel.gov/solar/old_data/nsrdb/1991-2005/tmy3/`

been designed as a first dataset for users new to microgrids. It contains 10 microgrid with the same architecture (PV + battery + genset) and is mostly aimed to gain some intuition for what is happening in the simulation. In `pymgrid25`, all the possible architectures can be found over 25 microgrids. There are four microgrids with only a genset, three with a genset and a grid, nine with only a grid, and nine with a genset and a weak grid.

As we propose these collections of microgrids as a standardize test set, we also implement a suite of control algorithms as a baseline comparison. Specifically, we implement four algorithms: rule-based control, model predictive control (MPC), Q-learning, and decision tree (DT) augmented Q-learning. [16]. Table 1 present the results obtained by running this suite on `pymgrid25`. By examining the results, we see that MPC with a perfect forecast can be viewed as nearly optimal while the RBC can be viewed as a lower bound as it gives the performance achievable by a simple algorithm. The bold values indicate the best performing non-MPC algorithm.

Table 1: Numerical results on `pymgrid25`

Architecture	Metric (k$)	MPC	Rule-based	Q-learning	Q-learning + DT
All	Mean cost	11,643	19,265	389,234	**13,385**
	Total cost	291,086	481,636	9,730,870	**334,624**
Genset only	Mean cost	19,722	57,398	337,385	**24,777**
	Total cost	78,890	229,593	1,349,543	**99,109**
Grid only	Mean cost	8,150	**8,372**	383,105	8,524
	Total cost	73,352	**75,350**	3,447,945	76,718
Grid + Genset	Mean cost	19,107	**22,327**	480,107	22,376
	Total cost	57,322	**66,982**	1,440,322	67,130
Weak grid	Mean cost	9,058	12,190	388,118	**10,185**
	Total cost	81,522	109,711	3,493,059	**91,666**

As we can see in Table 1, there are wide variations in the performance of the algorithms. Q-learning performs poorly; however, DT-augmented Q-learning outperforms the RBC – the mean cost of RBC is approximately 44% greater than DT-augmented Q-learning – with most of the difference occuring in edge cases, where RBC performs poorly as it lacks algorithmic complexity. While the DT-augmented Q-learning can perform well, it is still fatally flawed by the necessity of using a discrete action space; this requirement reduces the scope of actions the Q-learner is able to learn, and it is exceedingly difficult to ensure that all possible actions are considered in any given discrete action space. In six of the microgrids, RBC outperforms the DT-augmented Q-learning; this suggests that the DT Q-learning may not be able to consistently exceed an acceptable performance lower bound. This issue generally arises in microgrids with only a grid or with a grid and a genset.

While the difference between the DT-augmented Q-learning and the MPC often appears to be marginal, it is useful to keep in mind that these 25 microgrids have loads on the order of megawatts; as a result, a 13% difference can account for $40 million in additional costs. To fight climate change and integrate more renewables, the technology would need to scale beyond the thousands of system deployed. A few percentage points gained could tremendously reduce the operating costs, thus increasing the importance to improve controller performance. RL being a promising solution to achieve this goal.

5 Conclusion and Future Work

pymgrid is a python package that allows researchers to generate and simulate a large number of microgrids, as well as an environment for applied RL research. We establish standard microgrid scenarios for algorithm comparison and reproducibility, and provide performance baselines using both classical control algorithms and reinforcement learning. In order to improve RL-based microgrid controllers, it is critical to have a universal and adaptable baseline simulator – a role which ***pymgrid*** fills. A promising new avenue, is to leverage data generated from multiple microgrids to either increase performance or adaptability.

Immediate plans include the addition of a wider suite of benchmark algorithms, including extensive state of the art reinforcement learning approaches. We also plan to allow for additional microgrid components, more complex use cases, and finer time resolution. In addition, we hope to incorporate

the ability to pull real-time data. Finally, functionality surrounding carbon dioxide usage and data is valuable in allowing for users to control for carbon efficiency, along with allowing the user to consider roles that carbon tariffs may play in the future of energy generation.

References

[1] DRL-for-microgrid-energy-management. *https://github.com/tahanakabi/DRL-for-microgrid-energy-management*.

[2] microgridRLsimulator. *https://github.com/bcornelusse/microgridRLsimulator*.

[3] MicroGrids. *https://github.com/squoilin/MicroGrids*.

[4] Rikiya Abe, Hisao Taoka, and David McQuilkin. Digital grid: Communicative electrical grids of the future. *IEEE Transactions on Smart Grid*, 2(2):399–410, 2011.

[5] Bishnu P. Bhattarai, Sumit Paudyal, Yusheng Luo, Manish Mohanpurkar, Kwok Cheung, Reinaldo Tonkoski, Rob Hovsapian, Kurt S. Myers, Rui Zhang, Power Zhao, Milos Manic, Song Zhang, and Xiaping Zhang. Big data analytics in smart grids: state-of-the-art, challenges, opportunities, and future directions. *IET Smart Grid*, 2(2):141–154, 2019.

[6] Henrik Bode, Stefan Heid, Daniel Weber, Eyke Hüllermeier, and Oliver Wallscheid. Towards a Scalable and Flexible Simulation and Testing Environment Toolbox for Intelligent Microgrid Control. pages 1–9, 2020.

[7] Andrea Bonfiglio, Massimo Brignone, Marco Invernizzi, Alessandro Labella, Daniele Mestriner, and Renato Procopio. A Simplified Microgrid Model for the Validation of Islanded Control Logics. *Energies*, 10(8), 2017.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. pages 1–4, 2016.

[9] Tom Brown, Jonas Hörsch, and David Schlachtberger. PyPSA: Python for power system analysis. *Journal of Open Research Software*, 6(1), 2018.

[10] Nigel Bruce, Rogelio Perez-Padilla, and Rachel Albalak. Indoor air pollution in developing countries: A major environmental and public health challenge. *Bulletin of the World Health Organization*, 78(9):1078–1092, 2000.

[11] Mostafa Farrokhabadi, Dimitris Lagos, Richard W. Wies, Mario Paolone, Marco Liserre, Lasantha Meegahapola, Mahmoud Kabalan, Amir H. Hajimiragha, Dario Peralta, Marcelo A. Elizondo, Kevin P. Schneider, Claudio A. Canizares, Francis K. Tuffner, Jim Reilly, John W. Simpson-Porco, Ehsan Nasr, Lingling Fan, Patricio A. Mendoza-Araya, Reinaldo Tonkoski, Ujjwol Tamrakar, and Nikos Hatziargyriou. Microgrid Stability Definitions, Analysis, and Examples. *IEEE Transactions on Power Systems*, 35(1):13–29, 2020.

[12] Mevludin Glavic, Raphaël Fonteneau, and Damien Ernst. Reinforcement Learning for Electric Power System Decision and Control: Past Considerations and Perspectives. *IFAC-PapersOnLine*, 50(1):6918–6927, 2017.

[13] Anderson Hoke, Rebecca Butler, Joshua Hambrick, and Benjamin Kroposki. Maximum Photovoltaic Penetration Levels on Typical Distribution Feeders. *IEEE Transactions on Sustainable Energy*, (July):1–14, 2012.

[14] Adrian Kelly, Aidan O'Sullivan, Patrick de Mars, and Antoine Marot. Reinforcement Learning for Electricity Network Operation. pages 1–22, 2020.

[15] R.H. Lasseter, Abbas Akhil, Chris Marnay, John Stephepns, Jeff Dagle, Ross Guttromson, A. Sakis Meliopoulous, Robert Yinger, and Joe Eto. The MicroGrid Concept. 2001.

[16] Tanguy Levent, Philippe Preux, Erwan Le Pennec, Jordi Badosa, Gonzague Henri, and Yvan Bonnassieux. Energy Management for Microgrids: A Reinforcement Learning Approach. *Proceedings of 2019 IEEE PES Innovative Smart Grid Technologies Europe, ISGT-Europe 2019*, 2019.

[17] Magdi S. Mahmoud, Nezar M. Alyazidi, and Mohamed I. Abouheaf. Adaptive intelligent techniques for microgrid control systems: A survey. *International Journal of Electrical Power and Energy Systems*, 90:292–305, 2017.

[18] Daniel E. Olivares, Ali Mehrizi-Sani, Amir H. Etemadi, Claudio A. Cañizares, Reza Iravani, Mehrdad Kazerani, Amir H. Hajimiragha, Oriol Gomis-Bellmunt, Maryam Saeedifard, Rodrigo Palma-Behnke, Guillermo A. Jiménez-Estévez, and Nikos D. Hatziargyriou. Trends in microgrid control. *IEEE Transactions on Smart Grid*, 5(4):1905–1919, 2014.

[19] Maxx Patterson, Narciso F. Macia, and Arunachala M. Kannan. Hybrid microgrid model based on solar photovoltaic battery fuel cell system for intermittent load applications. *IEEE Transactions on Energy Conversion*, 30(1):359–366, 2015.

[20] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-dupont, Natasha Jaques, Anna Waldman-brown, Alexandra Luccioni, Tegan Maharaj, Evan D Sherwin, S Karthik Mukkavilli, Konrad P Kording, Carla Gomes, Andrew Y Ng, Demis Hassabis, John C Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. Tackling Climate Change with Machine Learning. pages 1–97.

[21] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[22] JD Taft. Electric Grid Resilience and Reliability for Grid Architecture. (March):16, 2017.

[23] Leon Thurner, Alexander Scheidler, Florian Schafer, Jan Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. Pandapower - An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, 2018.

[24] José R. Vázquez-Canteli and Zoltán Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied Energy*, 235(October 2018):1072–1089, 2019.

[25] Dongxia Zhang, Xiaoqing Han, and Chunyu Deng. Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems*, 4(3):362–370, 2018.