Pymgrid tool – Implementation Document

1. Environment

Pymgrid tool has Environment parent class (Environment.py file). This class provide the environment that support for/similar to openAI gym.

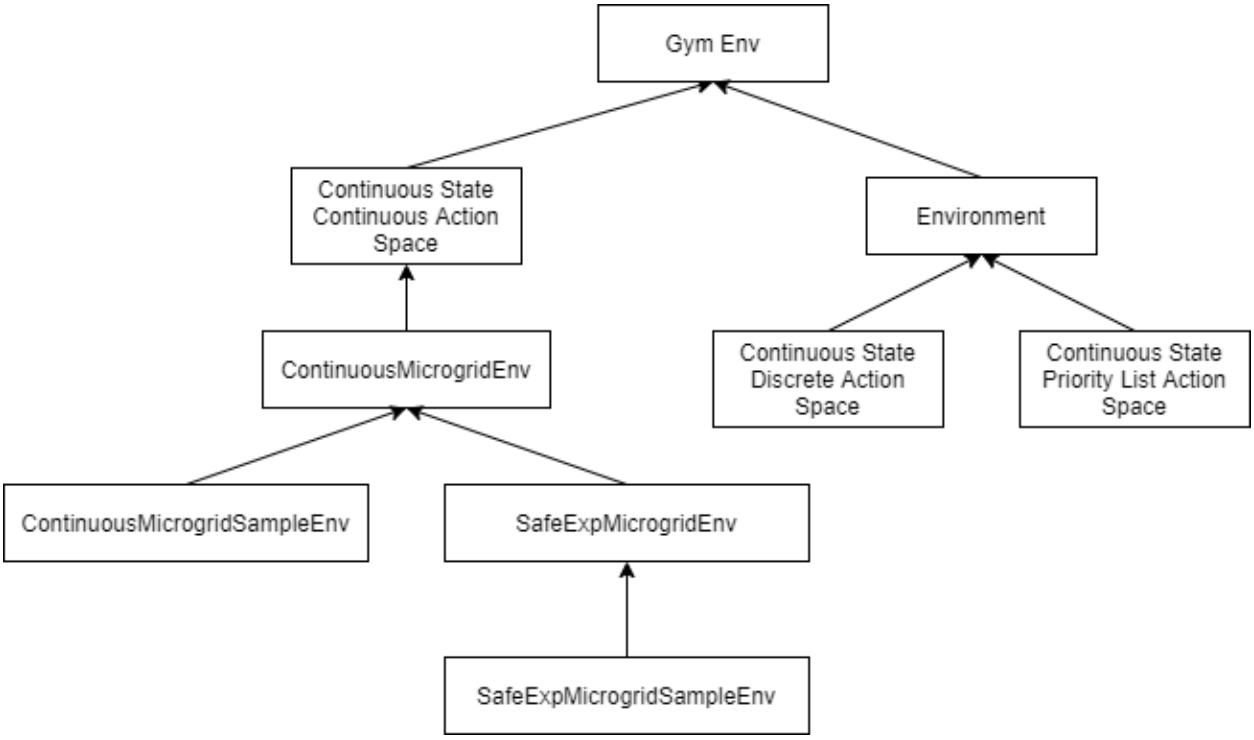


Figure – Hierarchy of class structure in Pymgrid Environments

2. Action space and state space

Pymgrid run method require control dictionary as parameter. (run method in pymgrid is similar to the step method in OpenAI Gym env)

Tool support for

1. Continuous state and discrete action space
2. Continuous state and continuous action space
3. Continuous state and priority list action space

2.1 Continuous state and discrete action space

Using pymgrid_csda.py file we can directly access the environment with continuous action space

$$From\ pymgrid.Environments.pymgrid\_csda\ import\ MicrogridEnv$$

```
state space:  Box(-1.0, inf, (11,), float64)
action space:  Tuple(Discrete(969), Discrete(525), Discrete(525), Discrete(2), Discrete(1920), Discrete(1920), Discrete(2))
```

Github link - https://github.com/anushaihalapathirana/RL-Pymgrid-tool/tree/master/discreteActionTuple

2.2 Continuous state and continuous action space

Using pymgrid_csca.py file we can directly access the environment with continuous action space $ContinuousMicrogridEnv$ class run a Microgrid in the format of gym environment with continuous states and continuous actions

$$From\ pymgrid.Environments.pymgrid\_csca\ import\ ContinuousMicrogridEnv$$

```
state space:  Box(0.0, inf, (10,), float64)
action space:  Box(-2.1646677449703104, 5.930005004993638, (5,), float64)
```

We can simply use this environment as below and apply RL algorithms as normal gym environment

$$env\ =\ ContinuousMicrogridEnv(microgrid)$$

This continuous state continuous actions class, $ContinuousMicrogridEnv$, has 2 child classes.

1. ContinuousMicrogridSampleEnv
2. SafeExpMicrogridEnv

2.2.1 ContinuousMicrogridSampleEnv

This class is inherited from ContinuousMicrogridEnv parent class but use samples generated from SampleAverageApproximation as states.

It generates new sample to use as load/pv/grid data and then call the parent reset function.

2.2.2 SafeExpMicrogridEnv

This class is inherited from ContinuousMicrogridEnv parent class but with constraint functionality for safety layer.

Constraints

- 2 constraints for Energy balance
- Constraint for power_charge
- Constraint for power_discharge
- Constraint for power_import
- Constraint for power_export
- 2 constraints for power generator set if genset is available in the microgrid architecture.

This SafeExpMicrogridEnv class has a child class, SafeExpMicroridSampleEnv, which is same as SafeExpMicrogridEnv except it use samples generated from SampleAverageApproximation as states.

Continuous action space Environment	**TD3 mean cost (10 episodes)**	**Std of cost**
ContinuousMicrogridEnv	2993878.11	235705.497
ContinuousMicrogridSampleEnv	2922777.06	199909.774
SafeExpMicrogridEnv	2926981.70	237709.307

Github Link – How to access continuous environments and apply TD3 algorithm - https://github.com/anushaihalapathirana/RL-Pymgrid-tool/tree/master/continuousActionSpace

2.3 3. Continuous state and Prioritize Action space

Using pymgrid_cspla.py file we can directly access the environment with prioritized list of discrete action space.

$$From\ pymgrid.Environments.pymgrid\_cspla\ import\ MicrogridEnv$$

This environment provides different discrete actions that agent can take. The number of actions is depends on the architecture of the grid. According to the action, the given control dictionary to the run() method is different.

Can access the action using get_action() method.

Possible actions

1. Battery charge – eg: only battery charge will change in the control dictionary
2. Battery discharge
3. Grid import
4. Grid export
5. Genset
6. Genset and battery discharge

We can also define actions as we want.

EXAMPLE:

```
If action == 2:
  Control_dict = {…# change the control dictionary parameters according to the action}
```

Usage of environment and how to apply RL algorithms to this environment can be found in section 7. AI Algorithms

3. State

State variables are,

- Current net_load (load - pv)
- Current battery capacity

4. Reward/Cost Methods

$$\boldsymbol{Reward = -Cost\ of\ operating\ the\ microgrid}$$

We can set the cost of co2 using set_cost_co2() method for a microgrid. Default co2 cost is 0.1

Cost calculation is based on below parameters.

1. Cost of loss load
2. Cost of overgeneration
3. Fuel cost (if genset available) – initialize to 0.4 in code level
4. Cost of grid export(-cost) and import(+cost) (if grid available) – these costs are based on existing tariff data
5. Cost of battery charge and discharge
6. CO2 cost

5. Co2 cost

Microgird class has internal method, _record_co2() to calculate the co2 cost of operating microgrid at each time step.

Co2 cost only change with the below actions. For other actions co2 cost is always zero

- Battery discharge
- Grid import
- Genset
- Genset and battery discharge

Approach 1 to optimize co2 cost only

I tried to set co2 cost as the reward and train the q learning algorithm with zero initial cost. It gave me zero co2 cost results.

Reason - When try to add co2 cost as reward, final testing results will always give zero cost because it always use battery charge action. Which is not calculate the co2 cost

Approach 2 to optimize co2 cost only

Setup an initial cost – setup 10 as initial cost (we can get an average cost and set it as initial cost)

Only consider the co2 cost as reward,

$$\boldsymbol{Reward = -Co2\ cost\ of\ operating\ the\ microgrid}$$

Changes in Pymgrid tool

- Introduced new data frame to record co2 cost and introduced 2 new methods in Pymgrid tool.

    1. get_co2_cost() – this function returns the co2 cost associated the operation of the each timestep. By calling this function, we can access the co2 cost.

    2. _record_co2_cost() – implemented this new internal method that record the co2 cost of operating the microgrid at each timestep

- Change run methods by setting co2 cost
- Reset the co2 cost dataframe in reset() method

Github link - changes in Pymgrid tool to optimize only co2 cost - https://github.com/anushaihalapathirana/pymgrid/tree/co2-cost-changes

Commit related to changes - https://github.com/anushaihalapathirana/pymgrid/pull/1/commits/7b76502029630628355383 2d94b2874fc3974b6a

5.1 How to optimize CO2 cost only

1. git clone https://github.com/anushaihalapathirana/pymgrid
2. cd pymgrid
3. pip install .
4. Then follow this example (dqn algorithm with co2 cost optimization) - https://github.com/anushaihalapathirana/RL-Pymgrid-tool/blob/master/co2_cost_optimization.py

6. Visualization

The tool provides several printing and visualization methods.

6.1 Plotting methods

- print_load_pv() – this method will plot load and pv data in input files
- print_actual_production() – plot actual production data
- print_control() – plot control dictionary data
- print_co2() – plot co2 data in input file
- print_cumsum_cost() – plot cumulative sum of cost
- print_cumsum_co2_cost() – Newly implemented as a part of co2 cost related changes. This function will plot the cumulative sum of co2 cost

6.2 Printing methods

- print_benchmark_cost() – print the cumulative cost of the different benchmark ran and different part of the dataset. Depending on if split it in train/test or not
- print_info() – print main information of the microgrid
- print_control_info() – prints the control_dictionary that use to control the microgrid
- print_updated_parameters() – print the latest values for the parameters of the microgrid changing with time

7. AI Algorithms

7.1. Q-Learning

Environment – Environments.pymgrid_cspla

Microgrid changes

- change the horizon (in hour) using set_horizon() method – this horizon is considered to control the microgrid. Use this when creating the q_table to get the forecasting values.
- Cost of co2 set to 0.5 using set_cost_co2() method – this will set the co2 cost of operating the microgrid at each step. Default = 0.1.

Initialize q_table as a dictionary.

state actions

{(net_load, state of capacity): {0: 0, 1: 0, 2: 0, 3: 0}, …}

Other hyperparameters

Number_of_episodes = 1000

Learning_rate = 0.01

Discount_rate = 0.9

Exploration_rate = 1

Update q value suing bellman equation and update exploration rate using epsilon greedy algorithm.

Output

Comparison of costs with

- pymgrid provided cost function
- newly implemented cost function which only consider the co2 cost
- modified the default cost function by removing the co2 cost from default cost *(consider only loss load, overgeneration, Fuel cost, grid export/import, battery charge / discharge costs)*

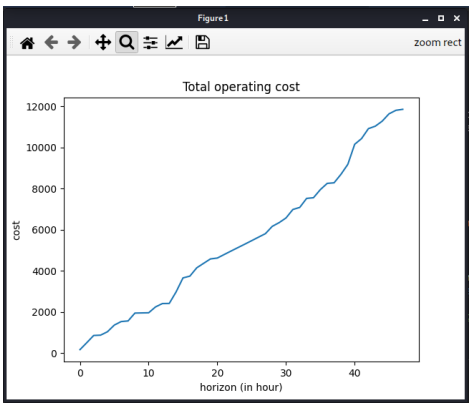
Note – co2 cost calculations are starting from 10

Architecture (48 hours)	**Default cost (€)**	**Only CO2 cost (€)**	**Default cost – co2 cost (€)**
PV, battery, grid	11 850.1	781.7	16 278.2
PV, battery, genset, grid	1 351 459.6	**7 709.9**	603 945.5
PV, battery, genset	**2 170 447.7**	480	**927 115.6**

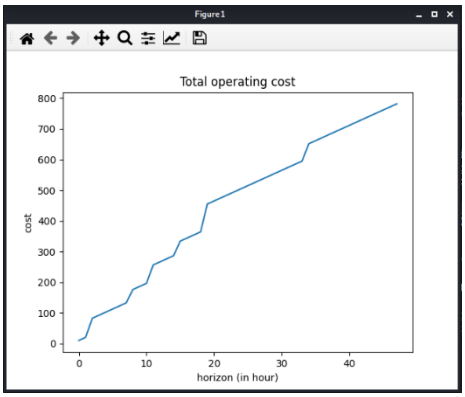
Total cost of the microgrid for given horizon.

1. Total cost of microgrid with PV, battery and grid architecture for 48 hours

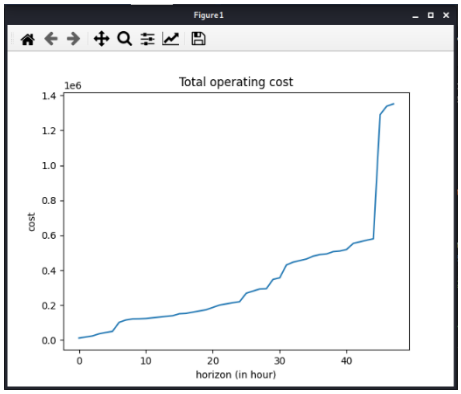
Default cost



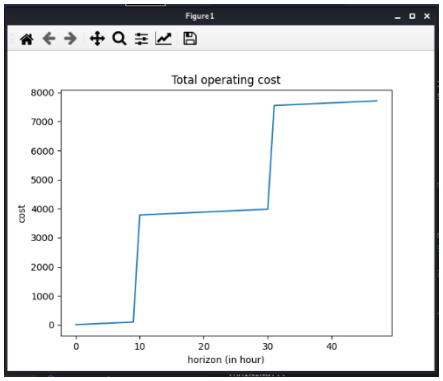
co2 cost



2. Total cost of microgrid with PV, battery, genset and grid architecture for 48 hours.

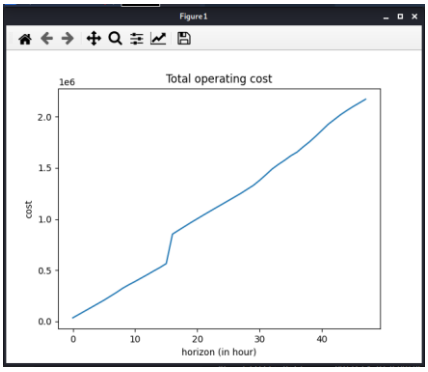


Co2

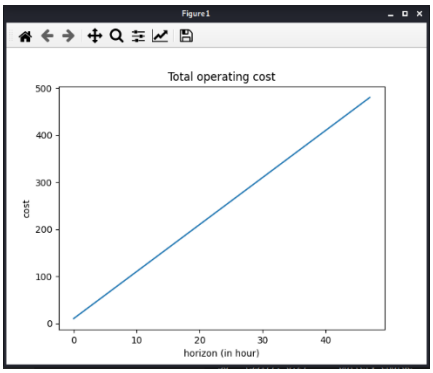


There are battery discharge and charge actions fired in the testing. That cause the cost gaps

3. Total cost of microgrid with PV, battery, and genset architecture for 48 hours.



Co2



Only battery charge action is fired in the testing period. Since this microgrid doesn't have grid, CO2 changes are not considered.

7.2. DQN

Environment – Environments.pymgrid_cspla

Microgrid changes

- Cost of co2 id set to 0.5

Configs

- Ray.rllib DQN default configs
- Batch mode = complete episode
- Env_config = microgrid

Training

Length of training process = 50 iterations

Add checkpoint to the best mean reward // or best max

Testing

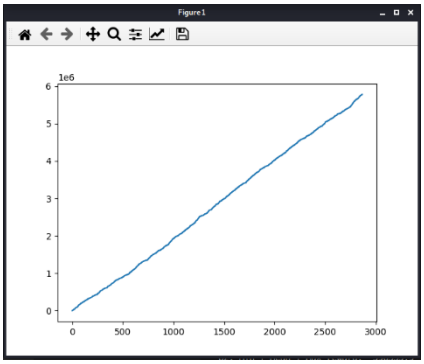
Restore the checkpoint and test using test data set.

Output

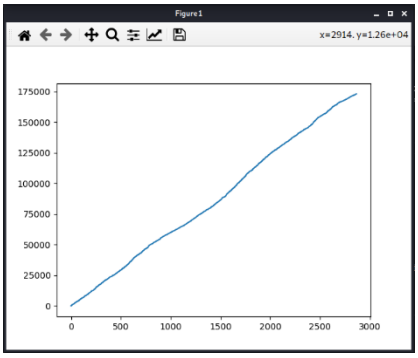
Total cost for operating microgrid at each timestep in test dataset

Total cost for operating microgrid at each timestep in test dataset is – 5 780 355.4753

Co2 cost – 173 047.3152



Co2



7.3. PPO

Environment – Environments.pymgrid_cspla

Microgrid changes

- Cost of co2 using set_cost_co2() method – this will set the co2 cost of operating the microgrid at each step. Default = 0.1- set it to 0.5

Configs

- Ray.rllib PPO default configs
- Hidden layers
- Batch mode = complete episode
- Env_config = microgrid

Training

Length of training process = 50 iterations

Add checkpoint to the best mean reward // or best max

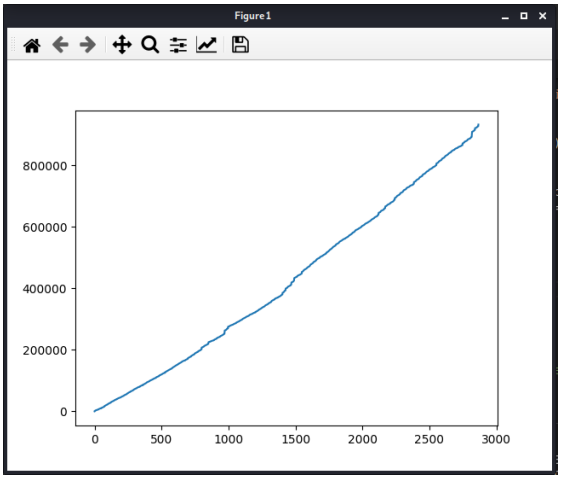
Testing

Restore the checkpoint and test using test data set.

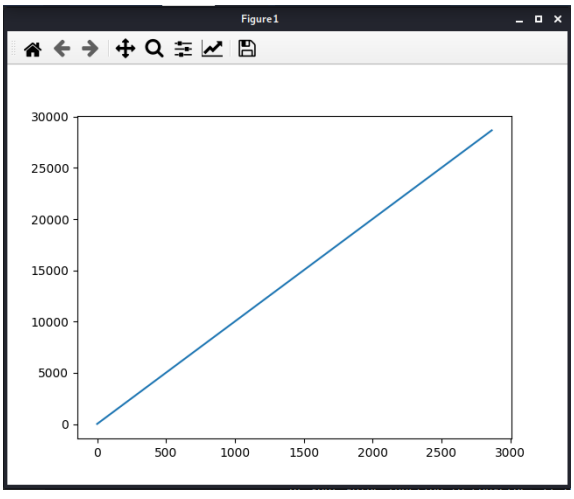
Output

Total cost for operating microgrid at each timestep in test dataset – 932 919.9703

Co2 cost – 28 670.0



Co2



7.4. Rule based control

Build in to the Pymgrid. Depending on the architecture of the microgrid and grid related import/export costs, pymgrid generates a priority list to be run in the rule based benchmark

Microgrid changes

- Set cost of co2 to 0.2

Output

1. architecture – PV, battery, grid

Cost of 8736 steps = 1 663 535.88

2. architecture – PV, battery, getset, grid

Cost of 8736 steps = 68 752 238.53

3. architecture – PV, battery, genset

Cost of 8736 steps = 163 343 861.34

7.5. Model Predictive control

Build in to the Pymgrid.

Prerequisites

Install cvxopt using pip install cvxopt.

Output

1. architecture – PV, battery, grid

Cost of 8736 steps = 1 537 632.77

2. architecture – PV, battery, getset, grid

Cost of 8736 steps = 64 219 416.49

3. architecture – PV, battery, genset

Cost of 8736 steps = 160 962 912.38

7.6. Sample average Approximation – version of stochastic mpc (sample average mpc control)

Build in to the Pymgrid.

Prerequisites

Install cvxopt using pip install cvxopt.

Output

1. architecture – PV, battery, grid

   Cost of 8736 steps = 1 552 635.35

2. architecture – PV, battery, genset, grid

   Cost of 8736 steps = 102 160 674.73

3. architecture – PV, battery, genset

   Cost of 8736 steps = 161 118 105.52

8. Results

8.1 Q leaning

Architecture (48 hours)	Default cost (€)	Only CO2 cost (€)	Default cost – co2 cost (€)
PV, battery, grid	11 850.1	781.7	16 278.2
PV, battery, genset, grid	1 351 459.6	**7 709.9**	603 945.5
PV, battery, genset	**2 170 447.7**	480	**927 115.6**

8.2 Other Algorithms – DQN and PPO

Co2 cost set to 0.5

Algorithm	Architecture	Default Cost on test data set	Co2 cost on test data set
DQN (50 training iterations)	PV, battery, grid	5 780 355.4753	173 047.31
	PV, battery, getset, grid	**448 657 414.5411**	30 025 640.67
	PV, battery, genset	329 309 664.5590	**48 728 718.26**
PPO (50 training iterations)	PV, battery, grid	932 919.9703	28 670.00
	PV, battery, getset, grid	45 287 292.8518	79 883.86
	PV, battery, genset	**199 632 680.9193**	**725 041.64**

8.3 Built in algorithms in Pymgrid tool

Co2 cost set to 0.2

Algorithm	Architecture	cost
Rule based control	PV, battery, grid	1 663 535.88
	PV, battery, getset, grid	68 752 238.53
	PV, battery, genset	**163 343 861.34**
Model Predictive Control	PV, battery, grid	1 537 632.77
	PV, battery, getset, grid	64 219 416.49
	PV, battery, genset	**160 962 912.38**
Sample Average Approximation	PV, battery, grid	1 552 635.35
	PV, battery, getset, grid	102 160 674.73
	PV, battery, genset	**161 118 105.52**

Note – Cost is high when microgrid contains generator set without a grid.

References

1. Pymgrid tool GitHub - https://github.com/Total-RD/pymgrid
2. Summer internship 2021 implementation Github - https://github.com/anushaihalapathirana/RL-Pymgrid-tool
3. Summer internship 2021 changes in Pymgrid tool to optimize only co2 cost - https://github.com/anushaihalapathirana/pymgrid/tree/co2-cost-changes

   Commit related to changes - https://github.com/anushaihalapathirana/pymgrid/pull/1/commits/7b765020296306283553832d94b2874fc3974b6a