# pycomm

Python reinforcement learning client for communication with the simulator.

## Usage

### Initialization: (Using Dynamic Optimization Environment as an Example)

```

from pycomm.pycomm import DynamicEnv

env = DynamicEnv(

channel\_type: int = 1, # Channel type

antenna\_type: int = 1, # Antenna type

job: str = 'job0', # Job name

output: bool = False, # Enable output

display\_micro: bool = True, # Macro and micro area selection

coverage\_range: int = 1000, # Coverage range

handover\_interval: int = 1, # Handover interval

)

```

### Simulator Step: Control the simulation steps of the simulator

#### Parameters:

- action: Actions provided by RL (Reinforcement Learning)

- start: Starting step of the simulator

- total: Total number of steps the simulator should execute (end step = starting step + total steps)

- interval: Time interval for each step

- is\_output: bool, whether to save visualization content.

#### Return Values:

- async\_coroutine\_step: Contains state and reward

- state: State returned by the simulator, used for RL optimization training

- reward: Result of taking action in the simulator, providing reward feedback

- visualize: Visualization Content

- terminated: Indicates when the simulation is finished

```

async\_coroutine\_step, terminated, \_ = await env.step(action, start, total, interval, is\_output )

```

### Closing the Simulator: Terminate the simulator process when RL training is complete.

```

await env.reset()

```

# RL Algorithm Interaction with Simulator

## Coverage Optimization RL Algorithm Interaction with Simulator

In this scenario, we divide the Beijing Guomao area into grids of 10m x 10m each. Using a channel model, we calculate the signal strength for each grid with respect to all base stations. We select the base station with the strongest signal for each grid and determine if the signal strength exceeds a predefined threshold. If it does, the grid is considered covered; otherwise, it is not covered. The objective of base station coverage optimization is to maximize the coverage of all grids and the total user throughput. The configuration of base station azimuth, downtilt, and beamwidth affects the propagation and reception quality of base station signals, thus impacting the coverage area. For example, when the base station's azimuth is small, the signal mainly propagates in the direction of the base station, leading to a smaller coverage area. On the other hand, a larger base station azimuth can result in signals propagating in a wider range but may also lead to signal dispersion and increased interference, resulting in weaker signal strength. The coverage optimization task involves optimizing the angles and beamwidth configurations of base stations to simultaneously improve the coverage rate and user total throughput. The state information for this task includes the status of each grid in the system, including signal strength from all base stations and whether it is indoors or outdoors. The reward function represents the reward signal received by the simulation engine based on feedback from different strategies, i.e., whether it is covered and the average communication rate per user. Based on the returned state and reward function, the network optimizer adjusts the azimuth, downtilt, and beamwidth configurations of base stations to optimize both the overall coverage rate and user total throughput of the grid.

The simulator transfers the current environment state (State) to the coverage optimization RL algorithm, which outputs actions (Action) and transfers them back to the simulator. The process continues in a loop until a termination state or condition is reached. The three elements of the coverage optimization RL algorithm, State, Action, and Reward, have the following structure:

State: [{"grid\_id": xxx, "signals": [xxx], "user\_number": xxx, "is\_indoor": true/false}] (Latest status for a given time period T)

Action: [{"id": xxx, "bs\_azi": xxx, "bs\_tilt": xxx, "bw\_azi": xxx, "bw\_tilt": xxx}]

Reward: [{"grid\_id": xxx, "coverage": true/false, "mean\_rate": xxx}] (Mean rate over time period T)

The meanings of each field are as follows:

State——

grid\_id: Grid identifier

signals: Signal strength from each base station to the grid

user\_number: Number of users in the grid (varies over time)

is\_indoor: Indicates whether it is indoor (affects signal strength)

Action——

id: Base station (antenna) identifier

bs\_azi: Azimuth of the antenna

bs\_tilt: Downtilt of the antenna

bw\_azi: Horizontal beamwidth (assumed as a constant)

bw\_tilt: Vertical beamwidth (assumed as a constant)

Reward——

grid\_id: Grid identifier

coverage: Indicates whether it is covered

mean\_rate: Mean rate over time period T (rate is related to received signal strength)

### Coverage Optimization RL Code User Guide

First, use the following command to activate the client and realize the link with the simulator:

./common\_docker config/……/common\_client/pycomm/pycomm/update\_config.yml -rl true

Open a new shell and activate the conda environment:

source activate torch-1.9.0-py37

Run the reinforcement learning program:

python Coverage\_simulator/train.py

After training is completed, three json files will be saved, namely reward\_his.json, cover\_his.json, and throughput\_his.json, which respectively save the reward value, coverage rate, and user data throughput value after each interaction with the simulator.

## Dynamic Optimization RL Algorithm Interaction with Simulator

Dynamic resource allocation in wireless networks aims to optimize base station access, resource block selection, and power allocation for certain time steps to improve user service quality and overall system performance. During the interaction, the simulation engine provides the optimizer with state and reward functions. The state space includes information about the system and users, such as user positions, remaining demands, and more. The reward function represents the reward signal received by the simulation engine based on feedback from different strategies, in this task, it is related to user throughput and satisfaction rate. Based on the returned state and reward function from the simulation engine, the network optimizer can choose corresponding strategies to apply to the simulation engine, evaluate their performance, including a series of actions such as user base station access, resource block selection, and power allocation.

The simulator transfers the current environment state (State) to the dynamic optimization RL algorithm, which outputs actions (Action) and transfers them back to the simulator. The process continues in a loop until a termination state or condition is reached. The three elements of the dynamic optimization RL algorithm, State, Action, and Reward, have the following structure:

State:

{

'uid\_list': List of user IDs,

'bid\_list': List of base station IDs,

'xy': User coordinates,

'unsatisfied\_demands': List of user demands (corresponding to the ID list),

'inv\_fading': Matrix of signal attenuation strength from base stations to users (corresponding to the ID list),

'base\_spectrum': List of base station spectra (corresponding to the ID list),

'base\_power': List of base station power (corresponding to the ID list),

}

Action:

[

{ "id": User ID,

"base\_stations": [[Accessed Base Station ID, Channel (1~45), Corresponding Transmission Power]]

},

…]

Reward:

[

{"flow": Throughput,

"satisfied\_flow": Satisfied demand=min{Demand, Throughput}

},

…]

### Dynamic Optimization RL Code User Guide

1. Run the simulator

. /comm\_docker -rl -config config.yml -listen "localhost:port\_xxx"

The port number "port\_xxx" of the listening address can be modified.

2. Train the model

python interfaces.py --mode train --port port\_xxx

The trained model is stored in the path of ". /model".

3. Evaluate the model

python interfaces.py --mode eval --port port\_xxx --n\_epoch yyy

The parameter "n\_epoch" refers to the number of epochs for which the model is trained.

## Energy-Saving RL Algorithm Interaction with Simulator

Energy-saving in wireless networks aims to reduce the energy consumption of the entire communication system by reasonably turning off some base stations and cells while ensuring user service quality. During the interaction, the simulation engine provides the optimizer with state and reward functions. The state space includes equipment parameters of base stations, user traffic demands, the affiliation between cells and base stations, and the coverage relationship between cells and users. The reward function represents the reward signal received by the simulation engine based on feedback from different strategies, in this task, it is related to the overall energy-saving rate of the system. Based on the returned state and reward function from the simulation engine, the network optimizer can choose corresponding strategies to apply to the simulation engine, including base station on/off states, cell on/off/sleep states, and the traffic demands on each cell.

The simulator transfers the current environment state (State) to the energy-saving optimization RL algorithm, which outputs actions (Action) and transfers them back to the simulator to obtain the current reward value (Reward). The three elements of the energy-saving optimization RL algorithm, State, Action, and Reward, have the following structure:

State:

{

'bs\_num': Number of base stations,

'cell\_num': Number of cells,

'demand':{

'user\_id': User ID,

'demand': User traffic demand,

},

'cell\_capacity': Cell capacity,

'bigraph\_cell\_bs': Affiliation between cells and base stations,

'bigraph\_demand\_cell': Coverage relationship between cells and users,

'air\_condition\_power\_coef': Relationship between air conditioning energy consumption and BBU+RRU energy consumption,

'bbu\_power': Energy consumption when BBU is on,

'cell\_sleep\_power': Cell sleep energy consumption,

'cell\_power\_coef': Relationship between energy consumption and load traffic when the cell is on,

}

Action:

{

'bs\_actions': Base station on/off states,

'cell\_actions': Cell on/sleep/off states,

'traffic': {

'user\_id': User ID,

'cell\_id': Cell ID,

'traffic': Load traffic

} // Traffic from user j to cell i

}

Reward:

{

'criteria':{

'power\_save\_ratio': Overall system energy-saving rate

}

}

### Energy-Saving Optimization RL Code User Guide

1. Run the simulator

. /comm\_docker -rl -config config.yml -listen "localhost:port\_xxx"

The port number "port\_xxx" of the listening address can be modified.

2. Train the model

python agent\_for\_simulator.py --mode train --port port\_xxx

The trained model is stored in the path of ". /output/mappo/".

3. Evaluate the model

python agent\_for\_simulator.py --mode eval --port port\_xxx