Reporting (based on final version)

1. Introduction

In this experiment, we use anaconda as the runtime environment and PyCharm as the development tool. We first designed a simplest prototype directly through PyGame, and after migrating to the gym framework we designed 5 progressively more complex versions.

The first version(tree_version_1) is a reproduction of the prototype(original), the second version (tree_version_1_Value_of_GHG)we consider the effects of GHG uptake, the third version(tree_version_1_2_weight_of_both_reward) we try to combine tree benefits and GHG effects into one value called reward, and the fourth version(tree_version_1.5) adds the concept of fertility, the fifth version incorporates all of the above, with code optimizations, parameters debugging work mainly during the fifth version (tree_version_2), and Docstring are written in fifth version.

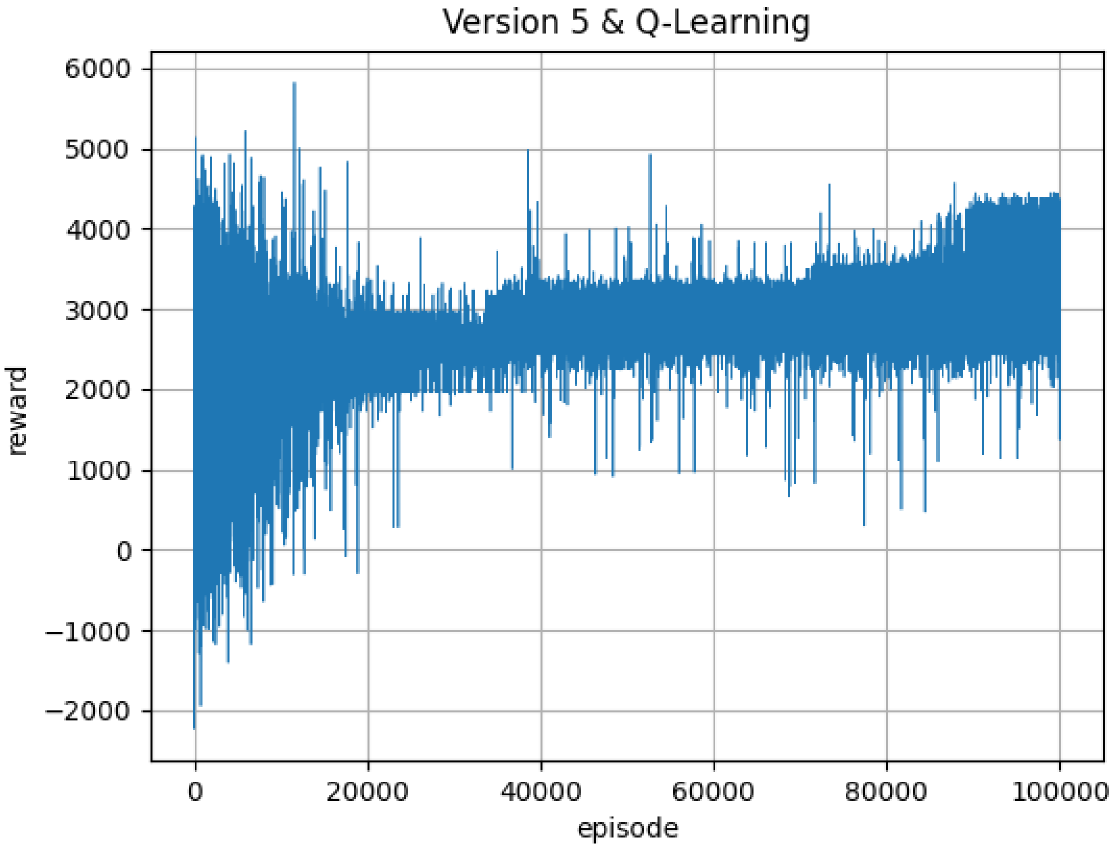
2. Setting

```
WEIGHT_TIMBER = 1.0
WEIGHT_GREENHOUSE_GAS = 0.05
""" two weights effect result of the reward """

MAX_FERTILITY = 3
""" the fertility of land will not more than 3"""

MINIMUM_REQ_GHG_10 = 200
"""
minimum requirement for Greenhouse gas in 10 year
People will protest if the minimum is not reached(get a negative reward)"""

MINIMUM_REQ_TIMBER_1 = 25
"""
minimum requirement for Timber in 1 year
If you don't meet the minimum you won't be able to pay the rent(get a negative reward)"""

RANDOM_SEED = 10
""" default of the random seed"""
```

reward function:

reward(weighted)=
*reward_timber * WEIGHT_TIMBER + Reward_greenhouse_gas *WEIGHT_GREENHOUSE_GAS*

3. Description of results



For q-learning, mean maintains a slow upward trend with very large fluctuations during this period.

4. Training and evaluation

Normally, we run 10,000 times during training, which takes about 2 minutes, and after adjusting some parameters, we run 100,000 times, which takes about 10 minutes.

5. Random seed

In this experiment we usually set the random seed as 10, and in several tests we will set the random seed as 0, 1 and 2.

5. Hyperparameters decisions

Q-Learning:
Hyperparameters: Learning rate: 0.2, Discount factor: 1.0, 3. Exploration rate: 1.0, Exploration rate decay: 0.9999, Minimal exploration rate: 0.05

DQN:
Hyperparameters: Learning rate: 1e-5, Discount factor: 1.0, Exploration rate: 0.9, Exploration rate decay: 0.9999, Minimal exploration rate: 0.05, Batch size: 64, Hidden layer: ReLU 100x100 ReLU, Optimizer: Adam, Replay buffer size: 5

Policy Gradient:
Hyperparameters: Learning rate: 1e-4, Discount factor: 1.0, Batch size: 128, Hidden layer: 256x64 sigmoid 64x256, Optimizer: Adam

6. Computer infrastructure

Song Zekun:
CPU:i7 6700HQ
GPU:GTX 1060
RAM:16 GB
OS: Windows 10

Zhao Huaxia:
CPU: i9-9900KF
GPU: 1660super
RAM: 32 GB
OS: Windows 10

Sui Guanqiao:
CPU: 2,3 GHz Quad-Core Intel Core i5
GPU: Intel Iris Plus Graphics 655
RAM: 16 GB 2133 MHz LPDDR3
OS: macOS Monterey

7. The average runtime for each result

CPU: i9-9900KF
GPU: 1660super
RAM: 32G
OS: Windows 10

Q_learning: 10000 rounds 77s
Deep_q_learning: 10000 rounds 102s
Policy_gradient: 10000 rounds 151s