**Multi-Agent Reinforcement Learning for Bus Control Optimization**

# Overview

Multi-agent reinforcement learning (MARL) for bus is an intelligent control system, designed to optimize bus dispatch at stops and reduce bus bunching on routes. It trains multiple agents (buses) to learn dynamic holding strategies using centralized training and decentralized execution. By integrating a simulation environment, centralized critic, and custom reward functions, MARL-Bus provides an adaptive, data-driven solution for enhancing bus service reliability.

# Key features

1. Reinforcement Learning Control

* Actor-Critic architecture for each agent (bus)
* Decentralized actor networks with centralized critic
* Adaptive and continuous holding time decisions

1. Simulation-based Environment

* Customizable multi-bus environment
* Realistic travel time and passenger modelling using sampled distributions
* Automatic overtake-prevention logic based on vehicle order

1. Reward Function Design

* r1: Headway deviation
* r2: Holding cost
* r3: Bunching penalty
* Smooth reward shaping using *tanh* and *sigmoid*; modifiable weights for flexible prioritization

# Technical Specifications

|  |  |  |
| --- | --- | --- |
| Critic/Actor parameters | | |
| Parameter | **Default Value** | **Description** |
| NUM\_AGENTS | 12 | The number of agents (buses) running in one route |
| STATE\_DIM | 4 | [stop ID, forward headway, occupancy, fleet order] |
| ACTION\_DIM | 1 | Holding level [0,1] |
| HIDDEN\_DIM | 256 | The number of neurons in the hidden layers |
| LR\_ACTOR | 1e-4 | Learning rate for the actor network optimizer. |
| LR\_CRITIC | 2e-4 | Learning rate for the critic network optimizer. |
| LR\_DECAY | 0.995 | Exponential decay factor for the learning rate scheduler. |

NOTE for states:

* Stop ID (INT): the ID of the stop at which bus is going to stop and make a holding decision.
* Forward headway (FLOAT): the time difference between the arrival times of two consecutive buses, in the range of [10, 2 \* TARGET\_HEADWAY]. (unit: second), , normalized by TARGET\_HEADWAY within range [-1,1].
* Occupancy (FLOAT): the number of onboard passengers divided by the capacity of bus, in the range of [0,1]
* Fleet order (INT): the sequential number in the fleet, that will be reorder after one loop.

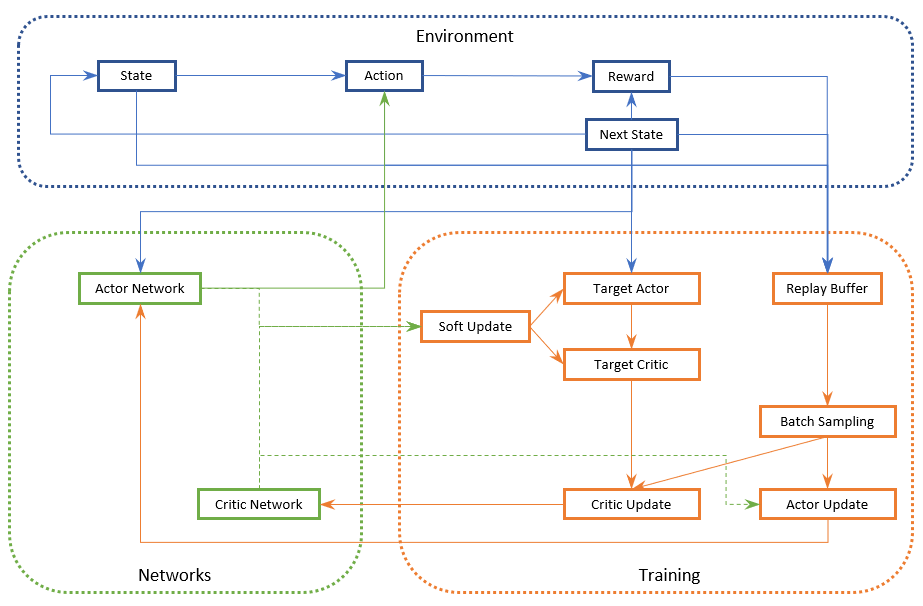
|  |  |  |
| --- | --- | --- |
| Environment parameters | | |
| Parameter | **Default Value** | **Description** |
| NUM\_STOPS | 44 (for A2386) | The number of stops in one route |
| TARGET\_HEADWAY | 600 sec | Ideal time gap between arrival of consecutive buses at one stop |
| MAX\_HOLD | 120 sec | The maximum holding time |
| BUNCHING\_THRESHOLD | 0.1\*TARGET\_HEADWAY | The headway threshold of bunching (if headway is low than it, bunching happens) |
| CAPACITY | 140 | The maximum number of passengers onboard |
| ALIGHT\_TIME | 2 | The alighting time per passenger |
| BOARD\_TIME | 3 | The boarding time per passenger |
| DOOR\_TIME | 5 | The time of closing/opening door |
| REST\_TIME | 120 sec | The rest time for at the final stop before restarting |
| REWARD\_SCALE  HEADWAY\_SCALE  HOLDING\_SCALE  BUNCHING\_SCALE | 2  1  0.2  0.8 | Scale factors for reward values |

|  |  |  |
| --- | --- | --- |
| Training parameters | | |
| Parameter | **Default Value** | **Description** |
| BATCH\_SIZE | 64 | Number of experiences (transitions) sampled from the replay buffer for each training step. |
| BUFFER\_SIZE | 100000 | Maximum number of past transitions stored in the replay buffer. |
| GAMMA | 0.995 | Discount factor |
| TAU | 0.005 | Soft update factor |
| MAX\_EPISODES | 400 | Training episode limit |
| NUM\_STEP | 6\*(len(stop\_id) + 1) | Training step limit |
| WARMUP\_EPISODES | 30 | The number of entire training episodes during which the agent collects experience without updating the networks (i.e., no learning). |
| WARMUP\_STEPS | len(stop\_id) + 1 | The number of initial time steps in each episode during which agents act but the system does not train. |
| EPSILON\_START | 1.0 | The initial threshold to explore |
| EPSILON\_END | 0.2 | The minimum threshold to explore |
| EPSILON\_DECAY | 0.995 | The decay rate of exploring strategy |
| GRAD\_CLIP | 0.5 | It limits the gradient norm of the actor/critic’s parameters during backpropagation. |
| WEIGHT\_DECAY | 1e-5 | A regularization technique that penalizes large model weights, to reduce overfitting and improve generalization. |
| MAX\_VALUE | 1e6 | It limits the range of values for safety and stabilization. |
| REWARD\_CLIP | 5.0 | It limits the single reward value for safety and stabilization. |
| PATIENCE | 50 | The maximum number of episodes without improvement |
| MIN\_EPISODES | 150 | The minimum number of training episodes |
| IMPORVEMENT\_THRESHOLD | 0.02 | The threshold of reward increment to determine improvement |

# System architecture

Actor-Critic Structure:

* Actors: output the action/policy based on state; aim at taking actions to maximize the Q-score from critic.
* Critic: assess the action of actor (i.e., Q-score); predict Q and real profit, and minimize it (i.e., make the assessment more accurate)



Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

# Training & Evaluation Workflow

1. Initialize environment and actor–critic networks
2. Warm-up phase: collect experience without learning
3. Experience gathering: agents interact with the environment
4. Training (**RL\_bus\_A2386.py**):

* Each agent trains its own actor
* All agents share a centralized critic to assess joint performance

1. Evaluation:

* Load saved actor policies
* Evaluate performance across all agents and identify the best

1. Inference/Application (**Call\_BestModel.py**):

* Load the actor class and the best model
* Input states of buses, and then generate action from model

# Outputs

The best policy models which can infer action values for bus based on the current state.

# Use Cases

1. Bus bunching mitigation for high-frequency transit corridors
2. Integration into smart bus fleet control systems
3. Reinforcement learning modules in academic or urban mobility simulators
4. Policy prototyping with real-time GTFS or SUMO-based simulations

# File Dependencies

|  |  |
| --- | --- |
| File | Description |
| travel time\_norm\_\*.xlsx | Travel time distributions based on 1-week data (Normal distribution)  Ordered by stop sequence of route  (History could be 1-month/1-year/more, to make it more precise) |
| route\_\*.json | The stops and segments of one route, which presents the sequence of stops. |
| Hist\_data.csv | The historical data based on each segment, including travel time, mean speed, etc. |

(\*: route id)

(The data processing process is done by **DataPrepare.py**)

# Deployment Notes

1. Built on PyTorch, GPU-supported
2. Modular environment for extensibility
3. Compatible with future integration into SUMO, GTFS, or OpenAI Gym-style frameworks