CSE 4/546: REINFORCEMENT LEARNING

Team 8: MULTI AGENT REINFORCEMENT LEARNING

Sri Amarnath Mutyala - : 50424956

Naga Kiran Reddy Karnati - 50432242

Pritisman Kar - 50388920

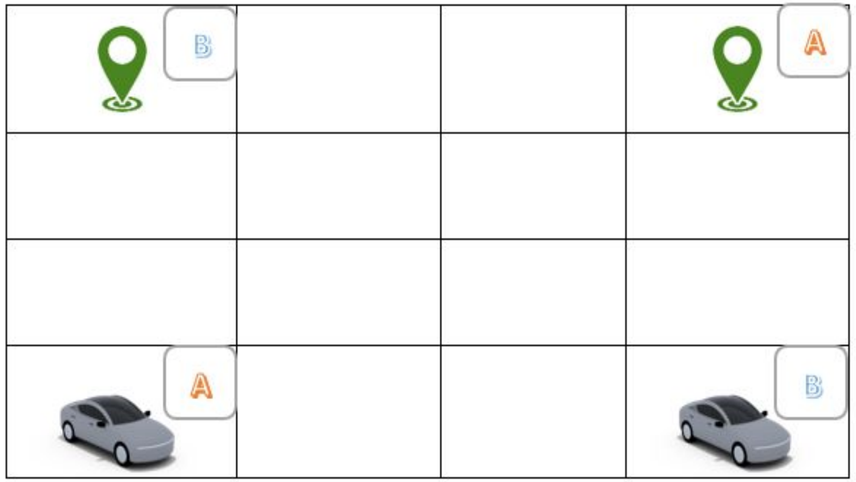
Academic Integrity Statement

"We certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that we did not receive any external help, coaching or contributions during the production of this work."

Contents

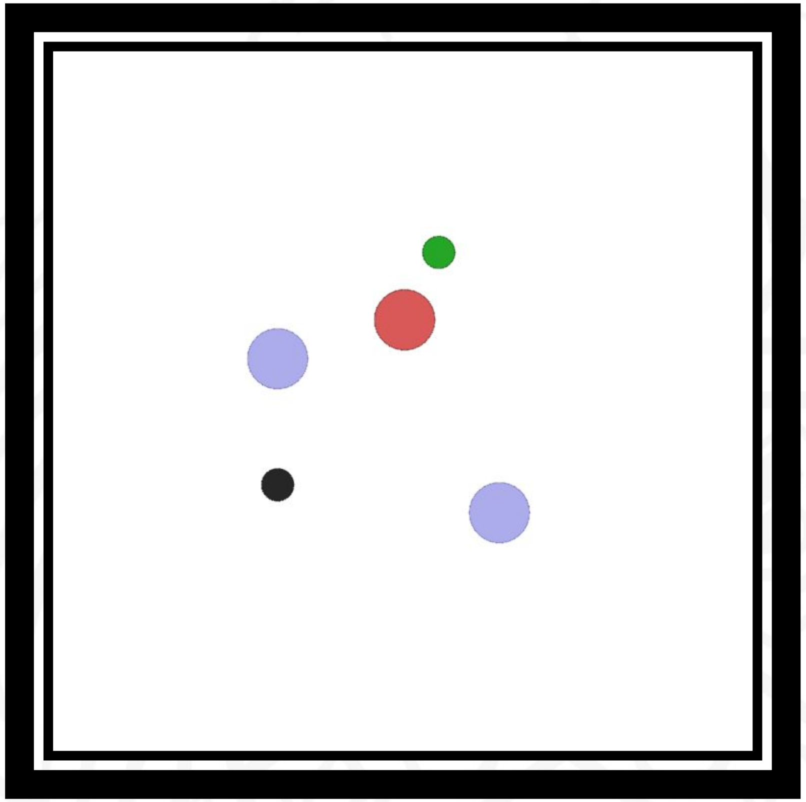
Vehicle Scheduling Environment – Multi-Agent Grid World (MAGW)

- Two cars in a **4x4** environment
- 1st car – Goal - To reach top right of the environment
- 2nd car – Goal - To reach top left of the environment
- State space: **16 states: {s0, s1, s2,...s15}**
- Action space: {0: down, 1: up, 2: right, 3: left, 4: no move}
- Reward structure
  - Towards the target: **1**
  - Away from the target: **-3**
  - Stays in same position: **-5**
  - Reaches target: **100**



Simple Adversary - OpenAI MA particle environment

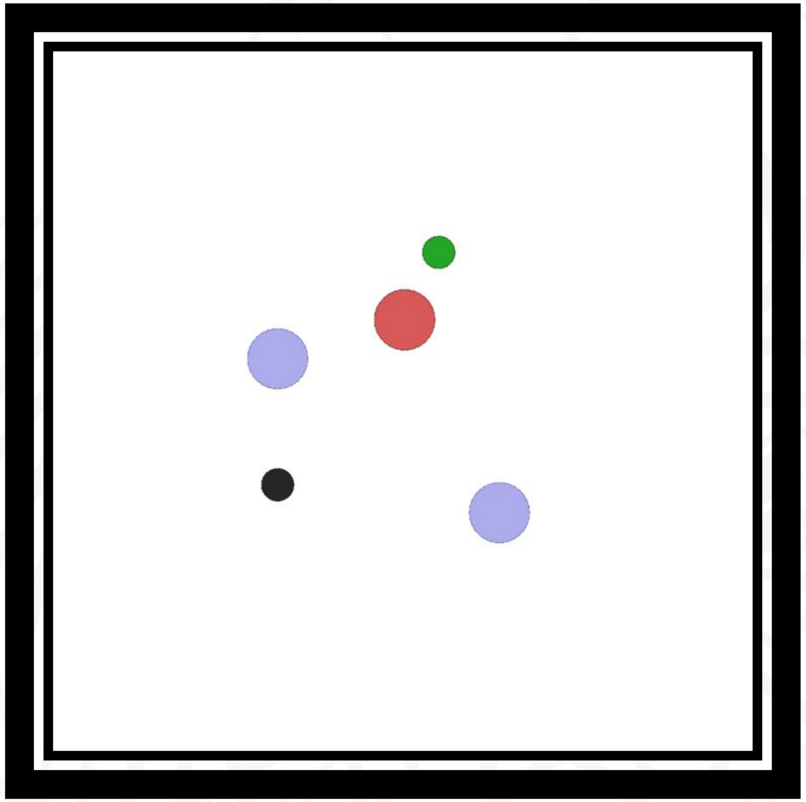
- 3 agents – **1** adversary, **2** good agents (Physical deception)
- Environment – 2 landmark (**Green** – target landmark, **Black** – dummy landmark)
- Rewards:
  - For agents:
    - **Positive** reward - based on distance between the closest agent to target landmark
    - **Negative** reward – based on distance between the adversary to target landmark
  - For adversary:
    - **Positive reward** – based on distance between the adversary to target landmark



https://www.pettingzoo.ml/mpe/mpe_simple_adversary.gif

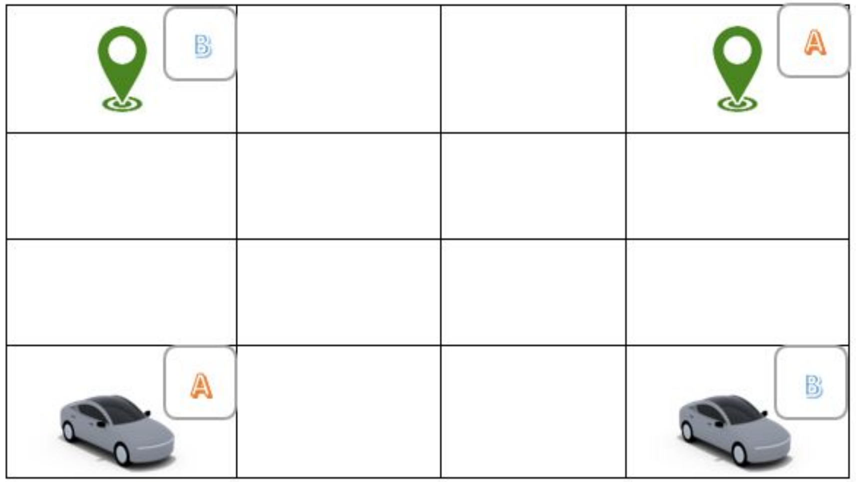
Simple Adversary - Main features

Parameter	Value
Actions	Discrete/Continuous
Agents	agents= [adversary_0, agent_0,agent_1]
Action Shape	(5)
Action Values	Discrete(5)
Observation Shape	(8) – adversary ,(10) – agents
Observation Values	(-inf,inf)
State Shape	(28,) [adversary + 2 agents]
State Values	(-inf,inf)

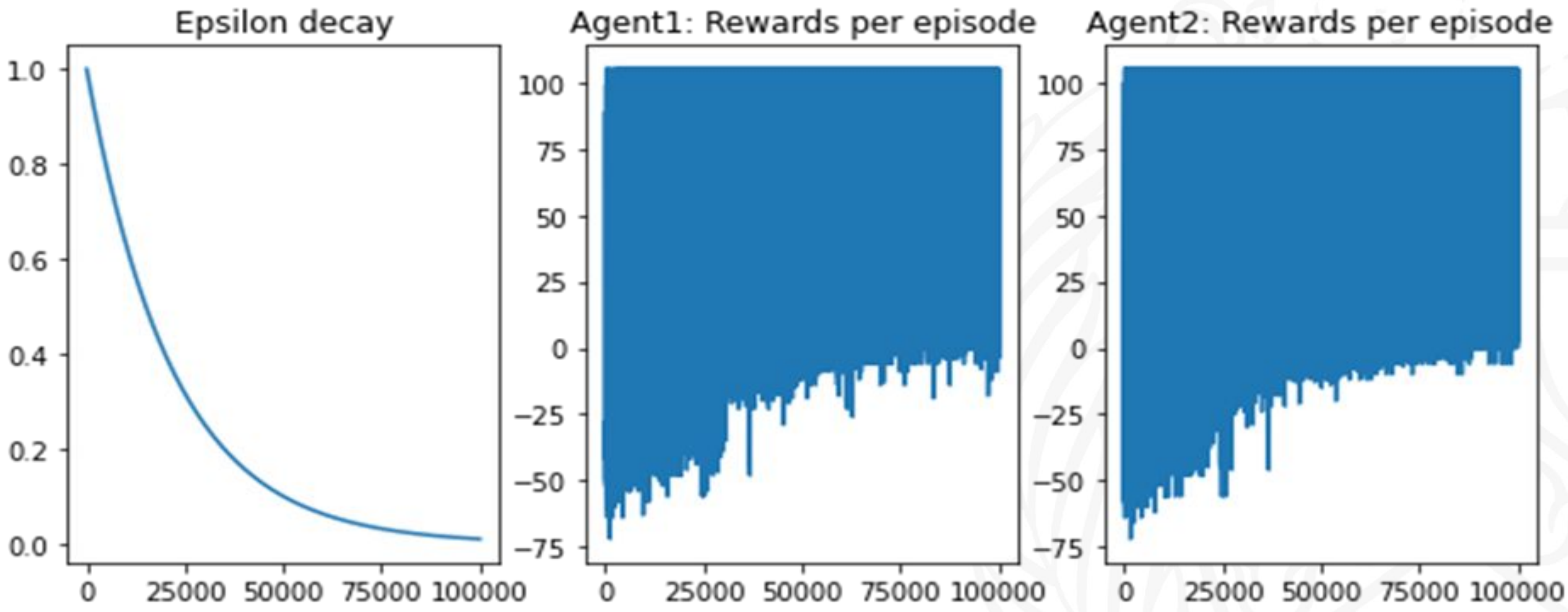


Q learning on Vehicle Scheduling Environment -MAGW

- Number of cars: **2**
- Episodes: **100000**
- Discount factor: **0.99**
- Learning rate: **0.15**
- Timesteps: **20**

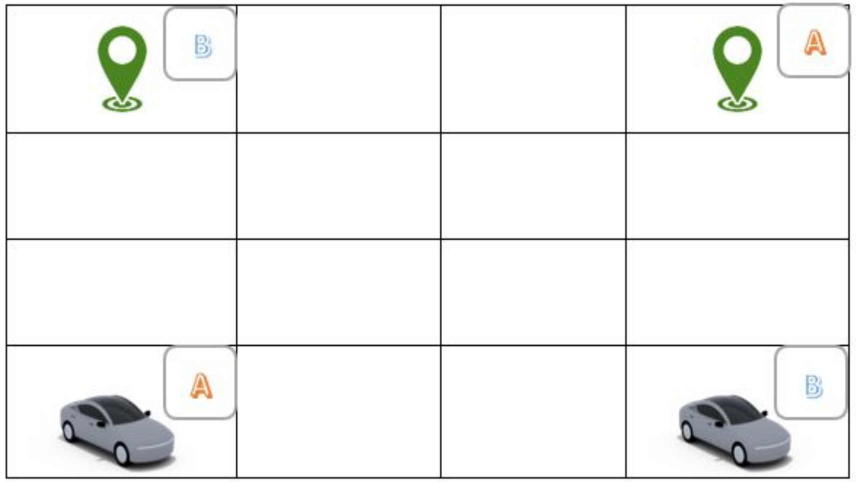


Results: Q-learning on MAGW



Results: Q-learning on MAGW

```
Agent 1 route:  [13, 9, 5, 6, 2, 3, 4]
Agent 2 route:  [16, 12, 8, 7, 6, 2, 1]
Agent 1 route:  [13, 9, 5, 6, 2, 3, 4]
Agent 2 route:  [16, 12, 8, 7, 6, 2, 1]
Agent 1 route:  [13, 9, 5, 6, 2, 3, 4]
Agent 2 route:  [16, 12, 8, 7, 6, 2, 1]
Agent 1 route:  [13, 9, 5, 6, 2, 3, 4]
Agent 2 route:  [16, 12, 8, 7, 6, 2, 1]
Agent 1 route:  [13, 9, 5, 6, 2, 3, 4]
Agent 2 route:  [16, 12, 8, 7, 6, 2, 1]
Agent 1 route:  [13, 9, 5, 6, 2, 3, 4]
Agent 2 route:  [16, 12, 8, 7, 6, 2, 1]
```



Multi Agent Deep Deterministic Policy Gradient

- First proposed by OpenAI in 2017
- Off-policy
- Algorithm which concurrently learns a Q-function and a policy
- Actor and Critic (i.e. and target networks for each) for each agent
- Can be used for environment with continuous action space and continuous environment
- Soft updates on parameters for each actor and critic network
- Uses Experience replay
- To ensure exploration, add Noise to deterministic policy gradient
- Issues – can overestimate the Q value in the critic network

Multi-Agent Deep Deterministic Policy Gradient Algorithm

For completeness, we provide the MADDPG algorithm below.

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma \, Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i(o_i^j)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  end for
end for

Implementation details:

Every Agent has

Actor Network:

Inputs: **States, actions**

Outputs: **Probs**

Critic Network:

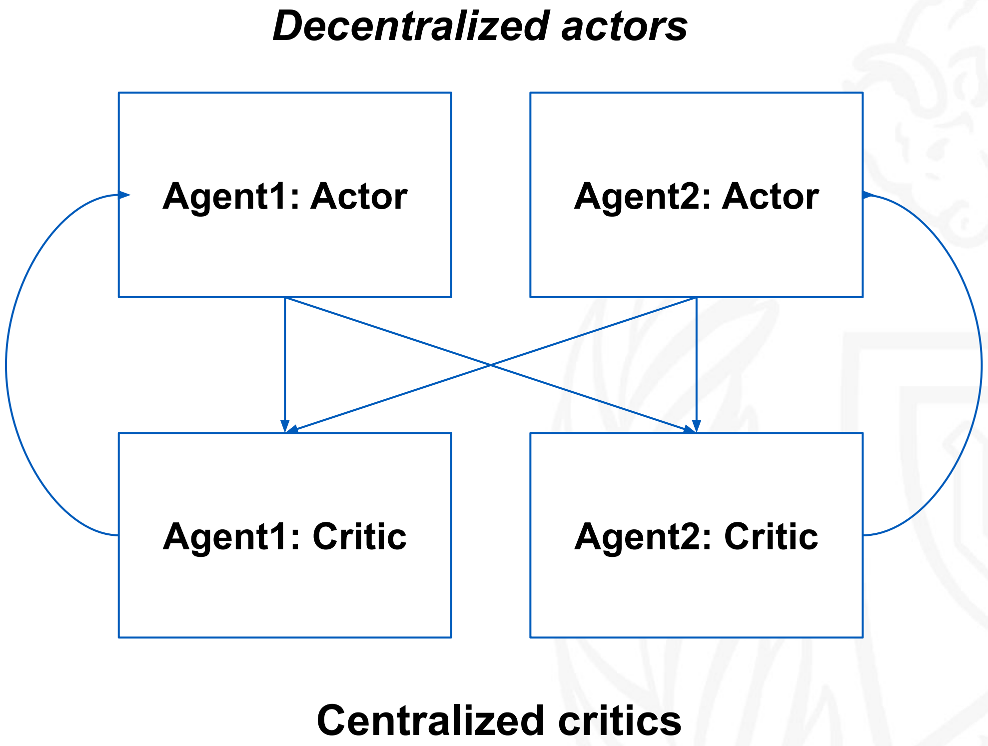
Inputs: **states, actions**

Outputs: **Q values**

To freeze weights to avoid running targets

Target Actor Network (i.e. performed soft updates)

Target Critic Network (i.e. performed soft updates)



MADDPG on MAGW

- Summary of actor network –

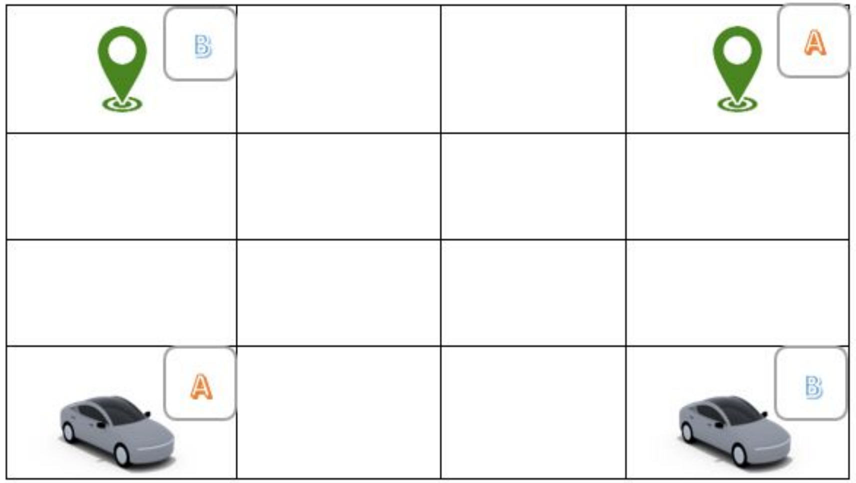
    - **2 hidden layers (64 each), softmax output, learning rate - alpha: 0.01**

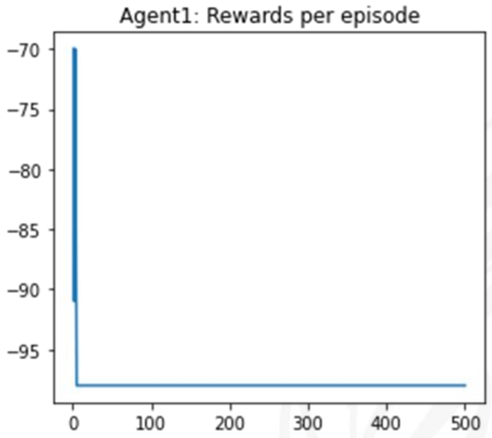
- Summary of the critic network –

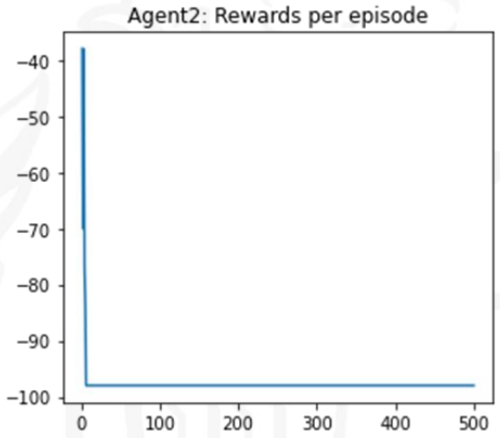
    - **2 hidden layers(64 each), relu output, learning rate - beta: 0.01**

- Number of episodes – **500**
- Gamma - **0.95**
- Batch size - **64**

Results: MADDPG on MAGW





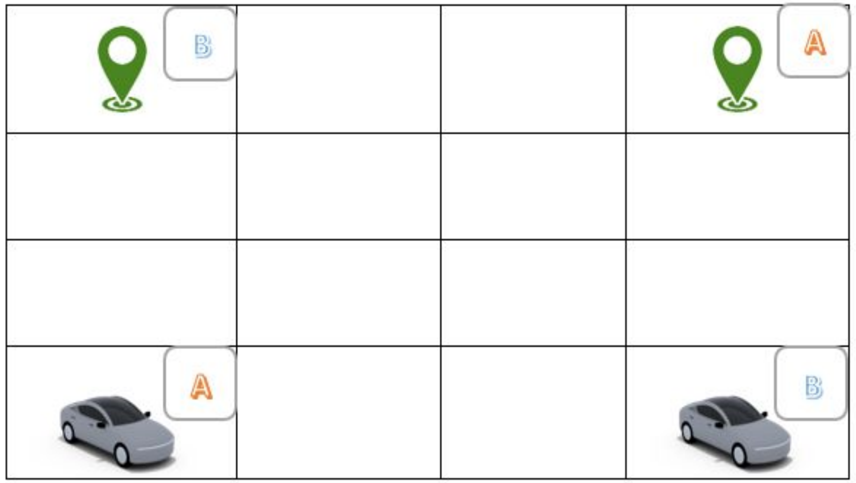


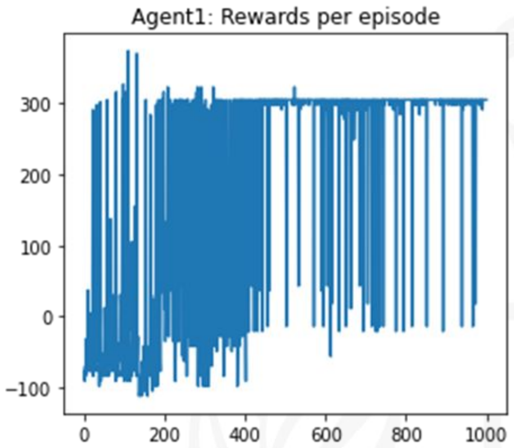

```
Observation:  Box(4, 4)
Actions:  Discrete(5)
Episode: 50, Individual Rewards: [-98, -98]
Episode: 100, Individual Rewards: [-98, -98]
Episode: 150, Individual Rewards: [-98, -98]
Episode: 200, Individual Rewards: [-98, -98]
Episode: 250, Individual Rewards: [-98, -98]
Episode: 300, Individual Rewards: [-98, -98]
Episode: 350, Individual Rewards: [-98, -98]
Episode: 400, Individual Rewards: [-98, -98]
Episode: 450, Individual Rewards: [-98, -98]
Agent1 Route:  [[3, 0], [2, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0], [1, 0]]
Agent2 Route:  [[3, 3], [2, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3], [1, 3]]
```

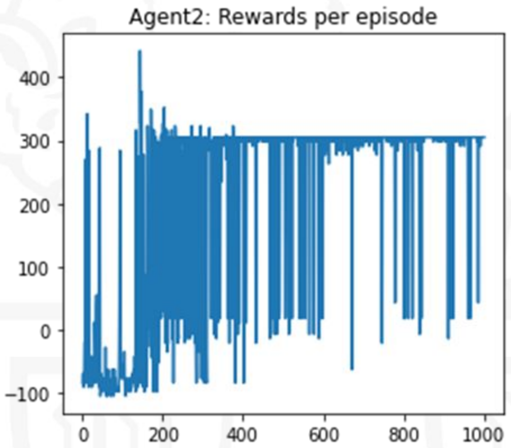
Improved MADDPG on MAGW

- **ε-greedy approach** even after applying noise to actions chosen from deterministic policy
- Number of episodes – **1000**
- Batch size: **128**
- Actor network: alpha: **0.001**
- Critic network: beta: **0.001**

Results: Improved MADDPG on MAGW








```
Observation:  Box(4, 4)
Actions:  Discrete(5)
Episode: 100, Individual Rewards: [-77, -77]
Episode: 200, Individual Rewards: [133, 327]
Episode: 300, Individual Rewards: [-70, 270]
Episode: 400, Individual Rewards: [-21, 305]
Episode: 500, Individual Rewards: [298, 298]
Episode: 600, Individual Rewards: [305, 305]
Episode: 700, Individual Rewards: [298, 298]
Episode: 800, Individual Rewards: [305, 18]
Episode: 900, Individual Rewards: [305, 305]
Agent1 Route:  [[3, 0], [2, 0], [2, 1], [1, 1], [0, 1], [0, 2], [0, 3]]
Agent2 Route:  [[3, 3], [3, 2], [3, 1], [2, 1], [1, 1], [0, 1], [0, 0]]
```

MADDPG on Simple Adversary - Implementation Details

- Summary of actor network –

  - **2 hidden layers (64 each), softmax output, learning rate - alpha: 0.01**

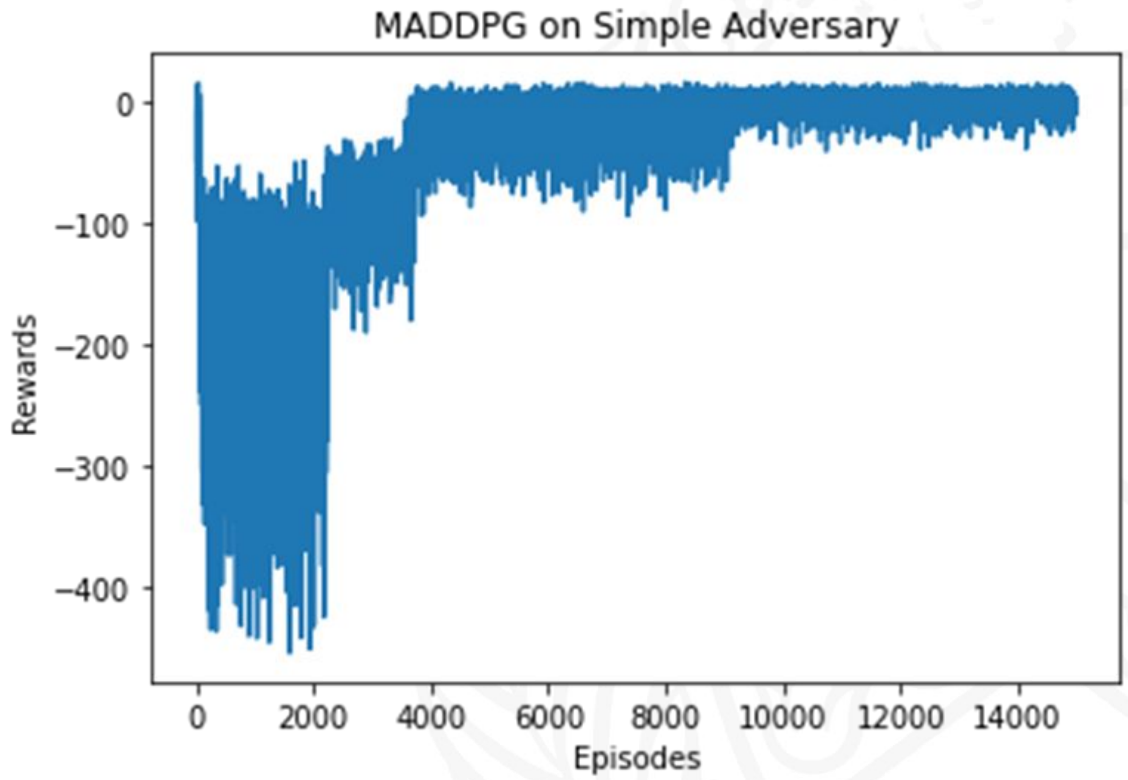
- Summary of the critic network –

  - **2 hidden layers(64 each), relu output, learning rate - beta: 0.01**

- Number of episodes – **15000**
- Gamma - **0.95**
- Batch size - **1024**
- Maximum timesteps: **25**
- Learning: **Every 100 steps**

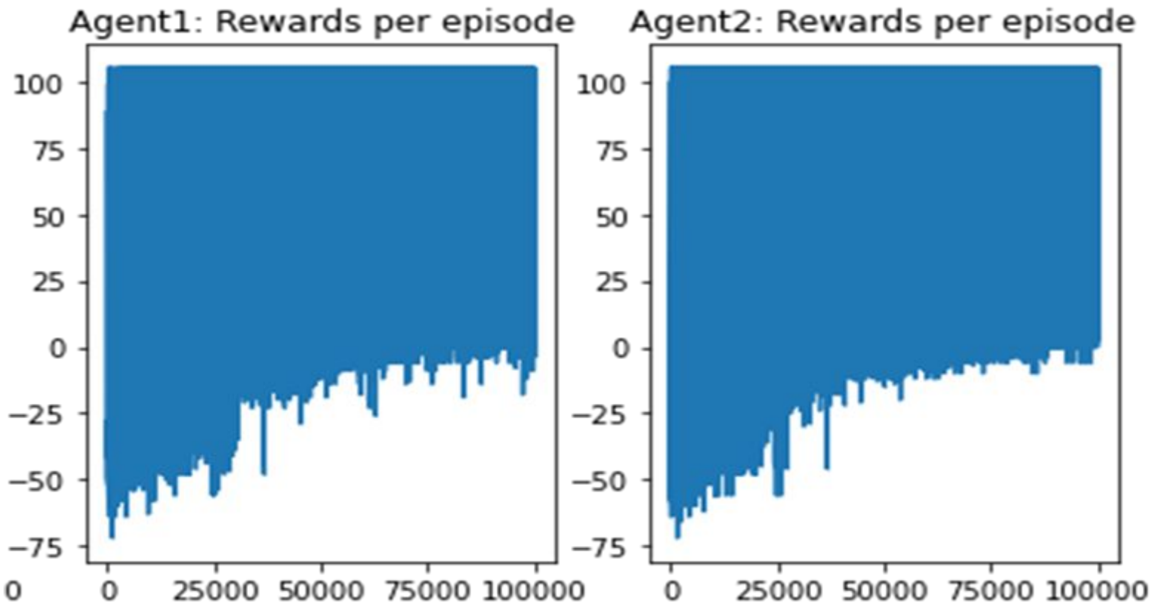
Results – MADDPG on Simple Adversary

```
episode 9850 average score 0.3
episode 9900 average score 1.2
episode 9950 average score 1.4
episode 10000 average score 2.3
episode 10050 average score 2.6
episode 10100 average score 1.6
episode 10150 average score 1.4
episode 10200 average score 1.3
episode 10250 average score 1.3
episode 10300 average score 1.8
episode 10350 average score 2.2
episode 10400 average score 2.4
episode 10450 average score 1.2
episode 10500 average score 0.9
episode 10550 average score 2.0
episode 10600 average score 1.8
episode 10650 average score 0.6
episode 10700 average score 0.9
episode 10750 average score 1.7
episode 10800 average score 0.7
```

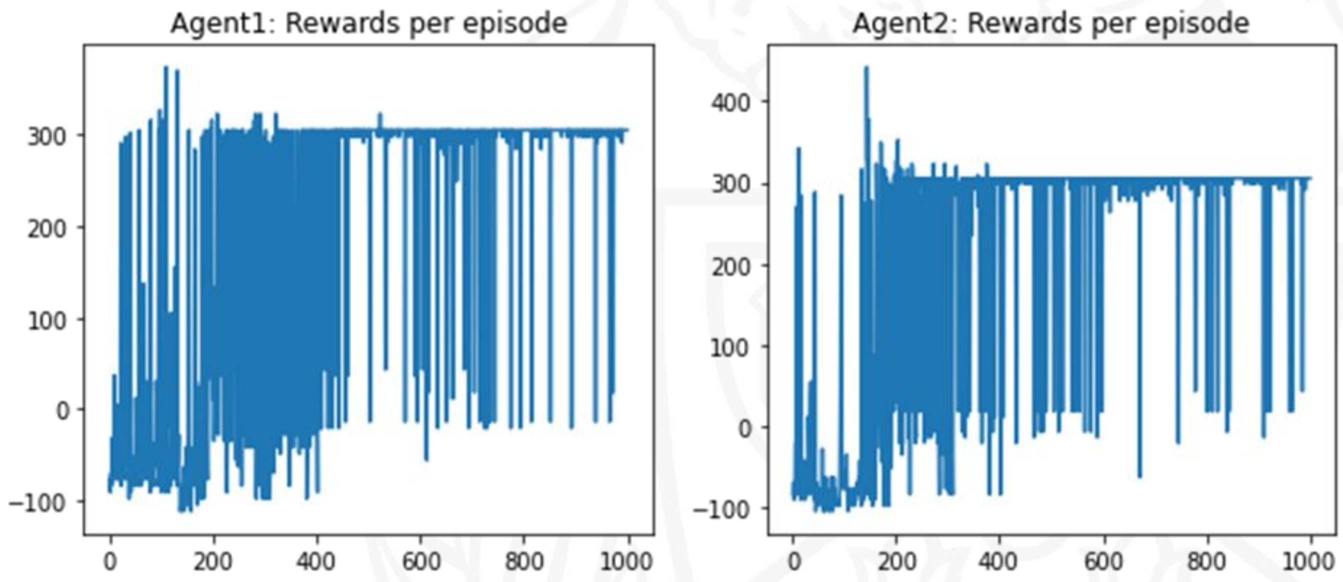


Comparison of Q-learning & Improved MADDPG on MAGW

Q-Learning



Improved MADDPG



Key observations

- As noticed in the previous slide, the Q learning is not working well for the Vehicle Scheduling environment.
- The MADDPG algorithm is working better when compared to Q learning algorithm.
- Proper attention should be given while implementing the MADDPG algorithm since it may lead to over-estimation of the Q-value using the Critic network.
- MADDPG is working well for continuous action-state value environment (Simple Adversary)

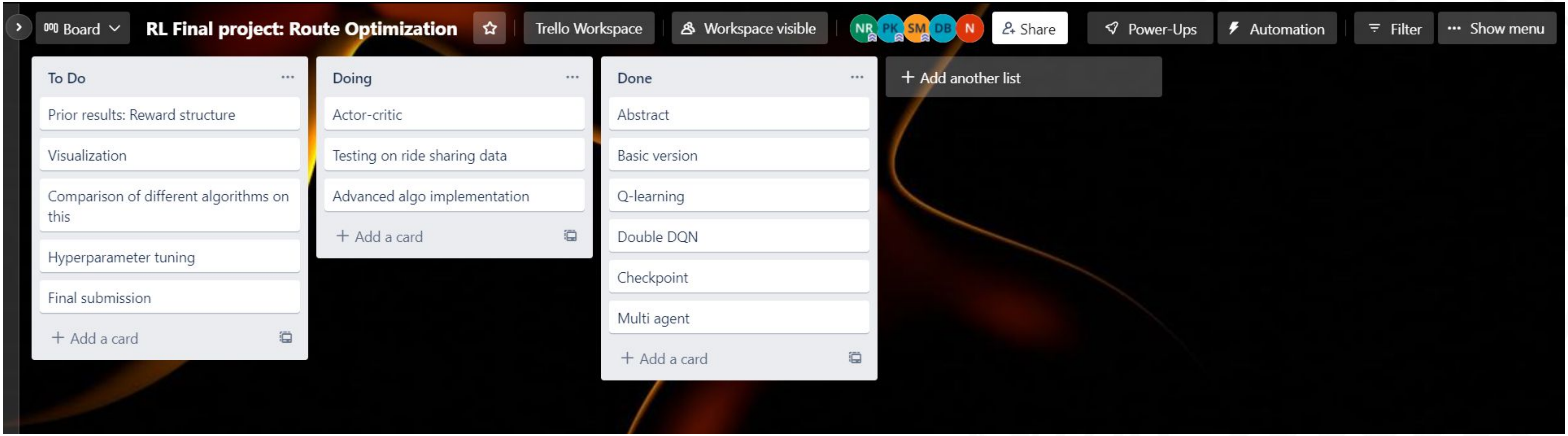
Other Trails & Extensions

- We were successful in setting up the Mujoco environment in CCR.
- We have also run benchmarks on Ant-v2 and HalfCheetah-v2 environments using RLLib package from the Ray Team.
- We eventually wanted to write our algorithm in the Ray tune framework as they have software primitives in RL and distributed computing.
- We have also defined a fleet scheduling environment with good reasonable assumptions.
- TODO: Apply the MADDPG algorithm to the fleet scheduling environment.

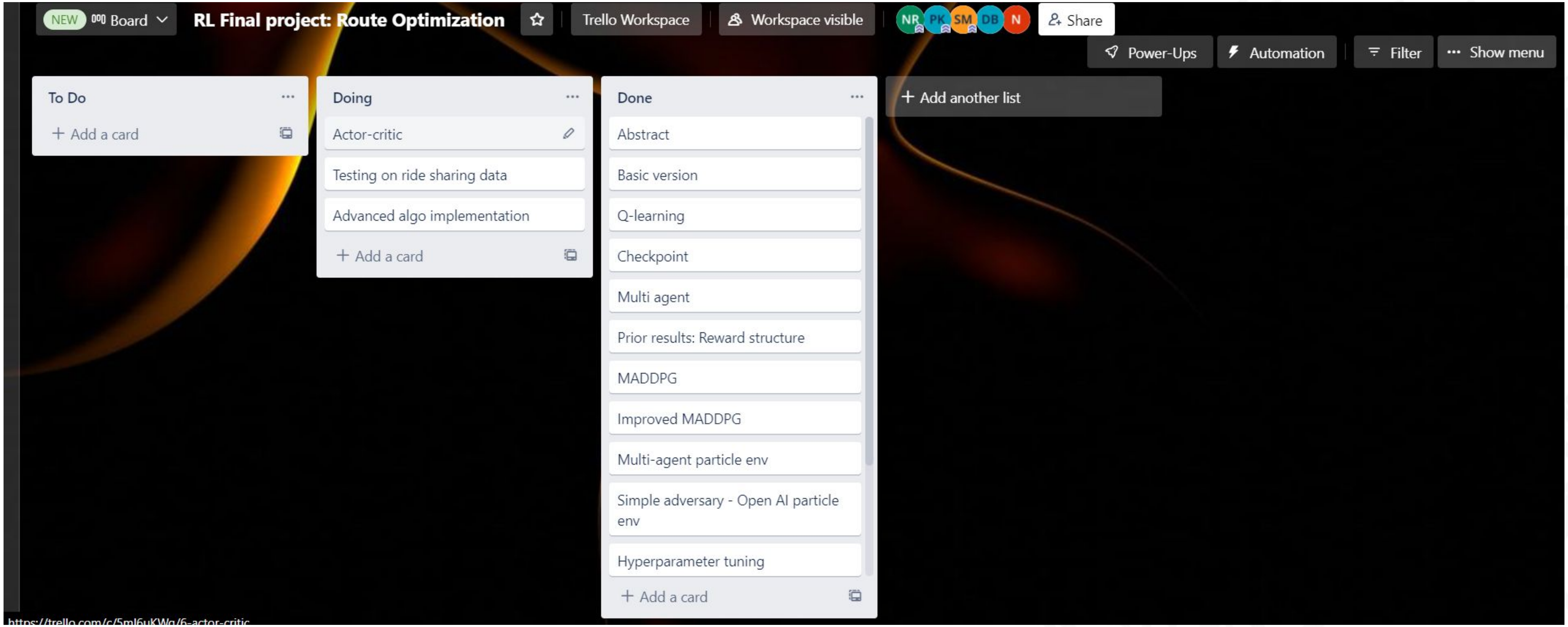
Contribution

Name	Contribution
Naga Kiran Reddy Karnati	33%
Sri Amarnath Mutyala	33%
Pritisman Kar	33%

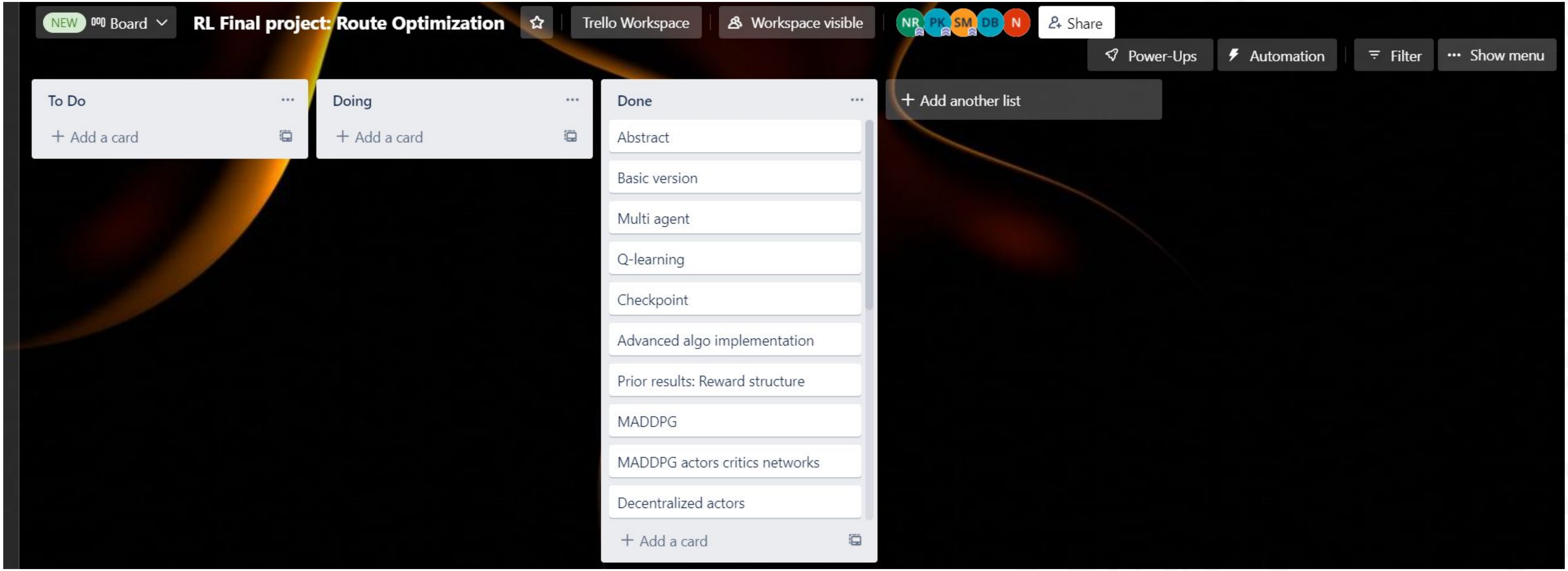
Trello workspace timeline



Trello workspace timeline



Trello workspace timeline



Reference

- 1706.02275.pdf (arxiv.org)
- GitHub - philtabor/Multi-Agent-Deep-Deterministic-Policy-Gradients: A Pytorch implementation of the multi agent deep deterministic policy gradients (MADDPG) algorithm
- https://www.pettingzoo.ml/mpe/simple_adversary.html
- Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning | Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining

THANK YOU