Learning Taxi Carpool Policies using MARL (Phase III)

COMP 597: Applications of Machine Learning in Real-World Systems

By Hung and Mushfique

Outline

- Recap
- Progress update
- Proposed Models
  - Reallocation with Contextual Actor-Critic (cA2C)
  - Cooperative Order Dispatching (COD)
  - Hybrid Model
- Project Plan
- Demo

Recap: Carpooling/Ride-Sharing Platform

- Taxis/ride-sharing platforms (Uber, Lyft) play significant role in daily commute
- **Problem:** Efficiently utilize road networks for
  - Minimum congestion
  - Optimal travel time and distance
  - Maximum profit
- **Applications:**
  - Dispatching orders, i.e. repositioning & matching driver to rider
  - *Route planning*
  - *Traffic signals control*
- **Goal:** MARL system that *maximizes profit, minimize travel time, distance* and *congestion*.

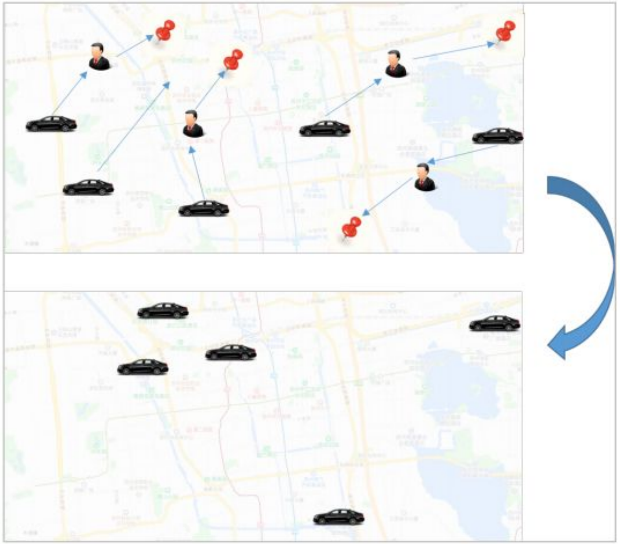


Figure 1: Order dispatching, driver repositioning & driver distribution after a certain timestep.

Recap: Proposed Models

- Reallocation with Contextual Actor-Critic (cA2C)
  - Lin, Kaixiang et al. “Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning”. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1774-1783. ACM, 2018.
- Cooperative Order Dispatching (COD)
  - Li, Minne et al. “Efficient Ridesharing Order Dispatching with Mean Field Multi-Agent Reinforcement Learning.” WWW (2019).
- Hybrid Model - Reallocation & Order Dispatching
  - Combining the benefits from both the previous models.

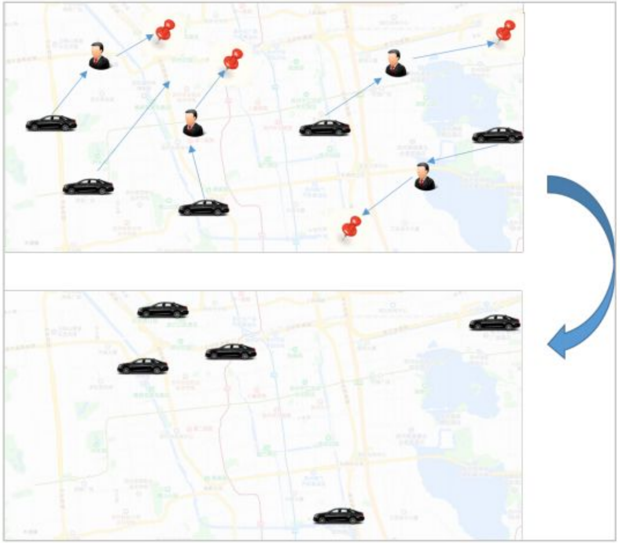


Figure 1: Order dispatching, driver repositioning & driver distribution after a certain timestep.

Progress Update

- Data exploration
  - [x] New York Taxi Dataset
  - [ ] *DiDi GAIA Open Dataset: no sign of approval*
  - [x] Visualization: setup city as hexagonal grid
- Research
  - [x] Understand methods in current literature
- Implement
  - [x] Implement cA2C
  - [~] Implement COD
  - [ ] Implement Hybrid Model
  - [x] Train on toy environment
  - [x] Train on real environment (NYC)
  - [ ] Compare results

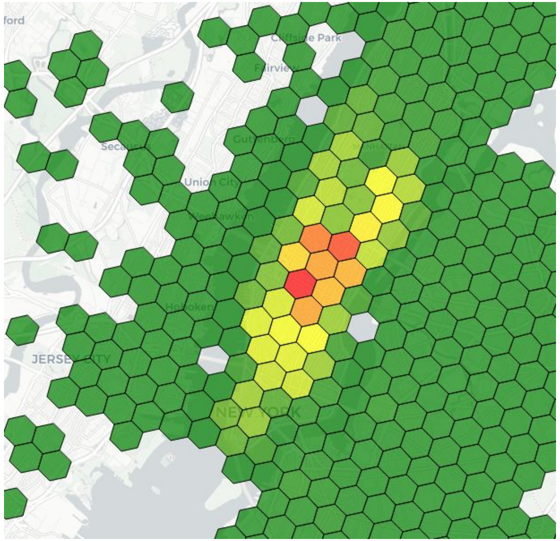


Figure 2: The grid-world visualization using NYC Dataset

Proposed Model 1:
Reallocation with Contextual Actor-Critic (cA2C)

Model Setup

- The fleet management problem is modelled as a **Markov Decision Process (MDP)/Markov Game, *G***
- MDP/Markov Game is defined by a tuple
  G = (N, S, A, P, R, γ)
- **N** = Number of Agents
- **S** = Set of States
- **A** = Joint Action Space
- **P** = Transition Probability Functions
- **R** = Reward Functions
- *γ* = Discount Factor

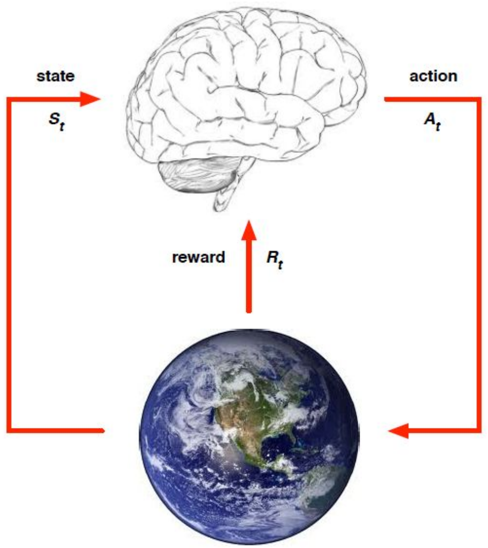


Figure 3: Markov Decision Process (MDP)

Model Setup

- **Environment:**
  - Hexagonal Grid world, 0.9km apart
  - Each Episode is represented by a day (24 hours), with 144 time steps (10 minute per step)
- **Agent (N):**
  - Each idle driver is an agent
  - Vehicles in the same grid node at a given time are homogeneous (treated like same agents)
  - Max. number of unique agents is N (total no. of grid nodes)

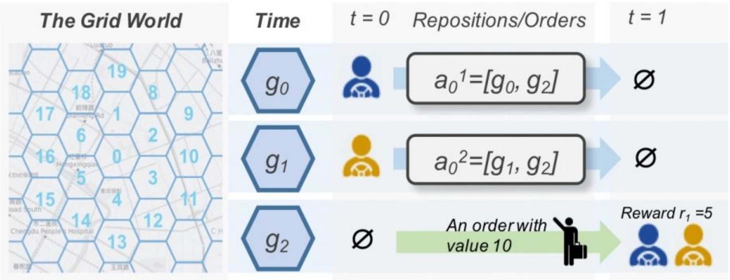


Figure 4: The grid-world system and a spatial-temporal illustration of the problem setting

Model Setup

- **State (S):**
  - Global state, s_t contains no. of agents (drivers) and orders for each grid node at time *t*.
  - Agent state, $s_t^i = [s_t , g_j]$, where g_j is one-hot-encoding of the grid ID.
- **Action (A):**
  - $A_i = [0,1,..6]$, 7 possible discrete actions, to six neighbouring grids, and the 7th to remain in the current grid.
  - $a_i^t = [g_0, g_1]$, representing movement from g_0 to g_1.
  - a_t is the joint action of all the N agents at time t.

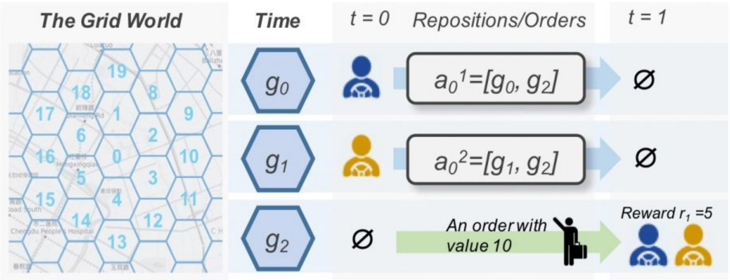


Figure 4: The grid-world system and a spatial-temporal illustration of the problem setting

Model Setup

- **Reward (R)/Discount Factor (ɣ):**
  - Individual reward, r^i_t is defined as the averaged reward of all agents arriving at the same grid node at the same time.
  - Reward is equal for agents in the same grid node g_j, $r_t(g_j)$.
  - Every agents tries to maximize its own expected discounted return, $\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r^i_{t+k}\right]$
- **State Transition Probability (P):**
  - Probability of arriving in a state s_{t+1} given the current state, s_t and the joint action vector, a_t.
  - $p(s_{t+1}|s_t, a_t) \sim [0, 1]$

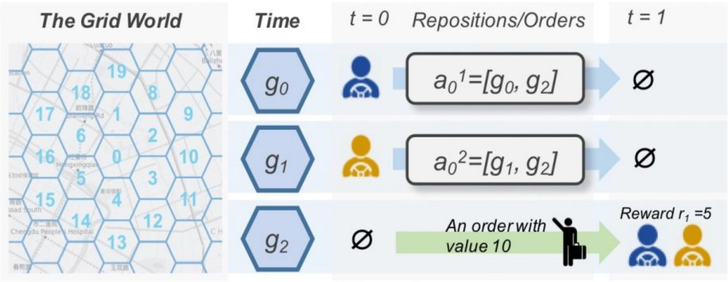


Figure 4: The grid-world system and a spatial-temporal illustration of the problem setting

Model Setup

Contexts:

- Geographic Context
  - Considers geographic nature of the place which can’t be accessed by agents. Ex - lakes, grid edges.

$$[\mathbf{G}_{t,\mathbf{g}_j}]_k = \begin{cases} 1, & \text{if } \mathbf{g}_d \text{ is valid grid,} \\ 0, & \text{otherwise,} \end{cases}$$

- Collaborative Context
  - Avoids the situation where agents move in conflict directions at a certain time for better performance. Ex - one agent going from g_1 to g_2, and another agent from g_2 to g_1.

$$[\mathbf{C}_{t,\mathbf{g}_j}]_k = \begin{cases} 1, & \text{if } Q(\mathbf{s}_t, \mathbf{g}_i) >= Q(\mathbf{s}_t, \mathbf{g}_j), \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathbf{q}(\mathbf{s}_t^i) = \mathbf{Q}(\mathbf{s}_t^i) * \mathbf{C}_{t,\mathbf{g}_j} * \mathbf{G}_{t,\mathbf{g}_j}.$$

Contextual Actor-Critic (cA2C)

- **Critic (Centralized):**
  - Centralized value function shared by all agents.
  - Function is updated by minimizing the loss function derived from Bellman equation:

$$L(\theta_v) = (V_{\theta_v}(\mathbf{s}_t^i) - V_{\text{target}}(\mathbf{s}_{t+1}; \theta'_v, \pi))^2,$$

$$V_{\text{target}}(\mathbf{s}_{t+1}; \theta'_v, \pi) = \sum_{a_t^i} \pi(a_t^i | \mathbf{s}_t^i)(r_{t+1}^i + \gamma V_{\theta'_v}(\mathbf{s}_{t+1}^i)).$$

- **Actor (Individual Agents):**
  - Uses Policy gradient to update its policies.

$$\nabla_{\theta_p} J(\theta_p) = \nabla_{\theta_p} \log \pi_{\theta_p}(a_t^i | \mathbf{s}_t^i) A(\mathbf{s}_t^i, a_t^i),$$

$$A(\mathbf{s}_t^i, a_t^i) = r_{t+1}^i + \gamma V_{\theta'_v}(\mathbf{s}_{t+1}^i) - V_{\theta_v}(\mathbf{s}_t^i).$$

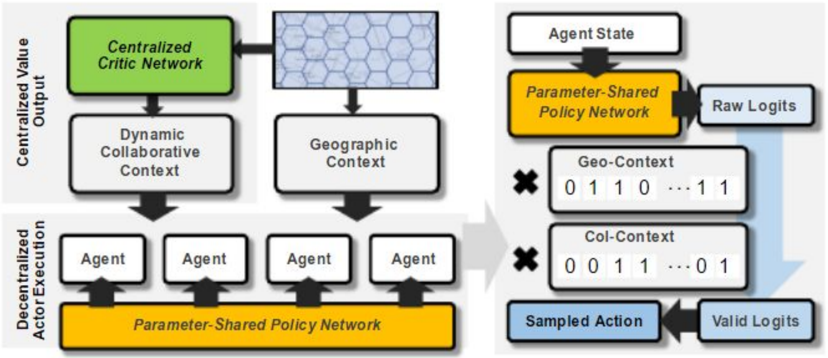


Figure 5: The coordination of **decentralized execution** is based on the output of **centralized value network**. The right part illustrates embedding context to policy network.

Simulator Design

1. Update vehicle status (setting some offline, and bringing some new vehicles online).
2. Generate new orders.
3. **Interact with the agents, passing the new global state to the fleet management algorithm (cA2C) and receive agent actions**
4. Reallocate the agents based on actions.
5. Assign available orders through a two-stage procedure:
   a. Orders in a given grid cell are assigned to agents in the same cell
   b. Remaining orders are assigned to vehicles in neighbouring cells.

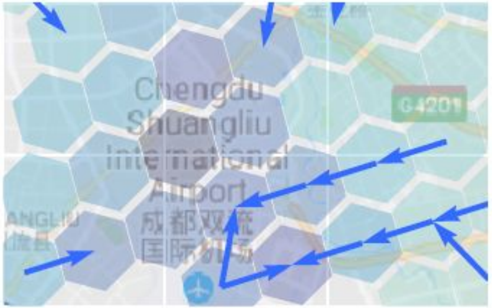


Figure 6: Illustration of the reposition by the model. Darker shade represents higher state value, and the blue arrows denote the repositions

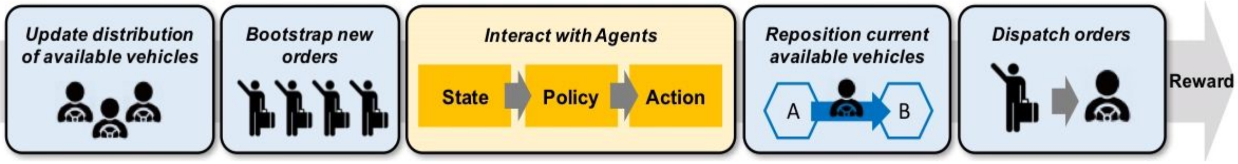


Figure 7: Simulator timeline in one time step (10 min.)

Proposed Model 2:
Cooperative Order Dispatching (COD)

Model Setup

- The order dispatching task is modelled as a **Partially Observable Markov Decision Process/Markov Game, *G***
- POMDP/Markov Game is defined by a tuple
  G = (N, S, O, A, P, R, γ)
- **N** = Number of Agents
- **S** = Sets of States
- **O** = Sets of Private Observations
- **A** = Joint Action Space
- **P** = Transition Probability Functions
- **R** = Reward Functions
- *γ* = Discount Factor

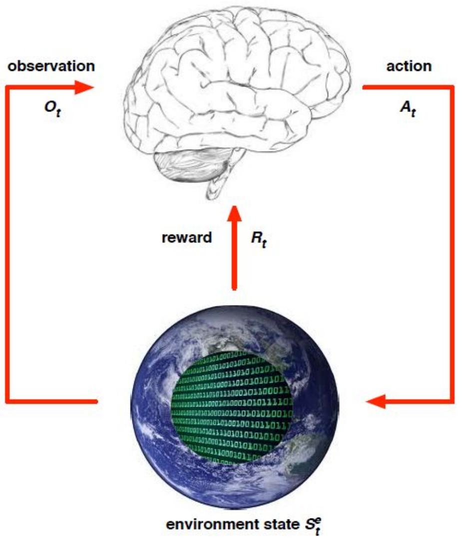


Figure 8: Partially Observable Markov Decision Process (POMDP)

Model Setup

- **Environment: (same as previous model)**
  - Hexagonal Grid, 0.9km apart
  - Each Episode is represented by a day (24 hours), with 144 time steps (10 minute per step)
- **Agent (N): (same as previous model)**
  - Each idle driver is an agent
  - Vehicles in the same grid at a given time are homogeneous (treated like same agents)
  - Max. number of unique agents is N (total no. of grid nodes)

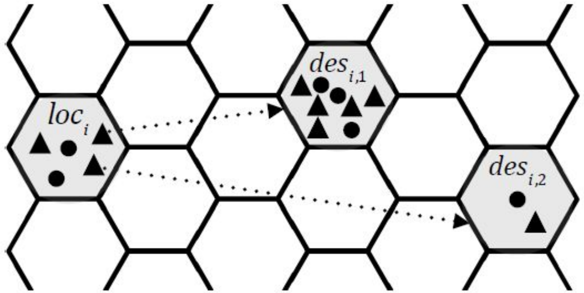


Figure 9: The grid-world system with agents (circles) and orders (triangles).

Model Setup

- **State (S)/Observation (O):**
  - Global state, s_t contains no. of agents (drivers), orders at time *t* and other environment dynamics (traffic congestion, weather conditions, etc.)
  - Agent observation, o_i contains its location loc_i, timestamp *t* and on-trip flag showing its availability.
- **Action (A):**
  - $a_{i,m}$ = [$des_{i,m}$], for i^{th} agent and m^{th} choice of order.
  - a_t is the joint action of all the N agents.
  - Use a deterministic policy, μ_i to generate ranking values of each observation-action pair (o_i, $a_{i,m}$) and use Boltzmann Softmax selector

$$\pi_i(a_{i,j}|o_i) = \frac{\exp(\beta\mu_i(o_i, a_{i,j}))}{\sum_{m=0}^{M_i} \exp(\beta\mu_i(o_i, a_{i,m}))}, \quad \text{for } j = 1, \ldots, M_i$$

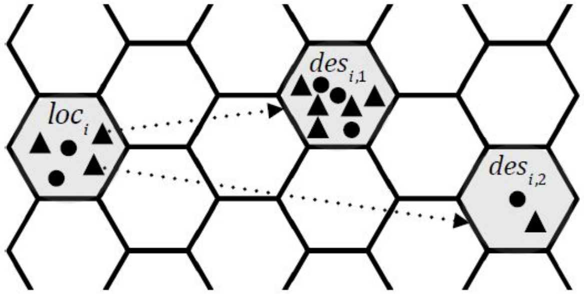


Figure 9: The grid-world system with agents (circles) and orders (triangles)

Model Setup

- **Reward (R)/Discount Factor (ɣ):**
  - Reward is defined as a combination of driver *i*'s own income ${}^{0}r$ and the order destination potential (DP) ${}^{1}r$ (only if #orders > #drivers).
    - *DP = #DD - #DS*
    - Difference in the demand (orders) and supply (drivers).
  - Each agent tries to maximize its own discounted reward, G^t.

$$r_i^t = {}^{0}r_i^t + {}^{1}\alpha\, {}^{1}r_i^t \qquad G_i^t = \sum_{k=t}^{\infty} \gamma^{k-t} r_i^k$$

- **State Transition Probability (P): (same as previous model)**
  - Probability of arriving in a state s_{t+1} given the current state, s_t and the joint action vector, a_t.
  - $p(s_{t+1}|s_t, a_t) \sim [0, 1]$

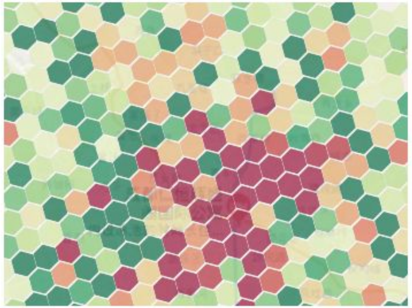


Figure 10: An example of destination potential (DP), with green grids representing more drivers than orders (and red if opposite), and the gap is proportional to the shade of colors.

Cooperative Order Dispatching (COD) with Mean Field (MF) Approximation

- **Critic (Centralized)**:
    - Each critic is trained by minimizing the loss function

$$\mathcal{L}(\phi_i) = \mathbb{E}_{\mathbf{o},\mathbf{a},\mathbf{r},\mathbf{o}'}[(r_i + \gamma\, v_{i^-}^{\mathrm{MF}}(\mathbf{o}_i') - Q_i(o_i, (\bar{a}_i, a_i)))^2],$$

$$v_{i^-}^{\mathrm{MF}}(o_i') = \sum_{a_i'} \pi_i^-(a_i'|o_i')\mathbb{E}_{\bar{a}_i'(\mathbf{a}_{-i}')\sim\boldsymbol{\pi}_{-i}^-}[Q_i^-(o_i', (\bar{a}_i', a_i'))],$$

    - v_i^{MF} is the Mean Field value function
- **Actor:**
    - Uses Policy Gradient to update its policies.

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathbf{o},\mathbf{a}\sim\mathcal{D}}[\nabla_{\theta_i}\mu_i(o_i, a_i)\nabla_{a_i} Q_i(o_i, (\bar{a}_i, a_i))].$$

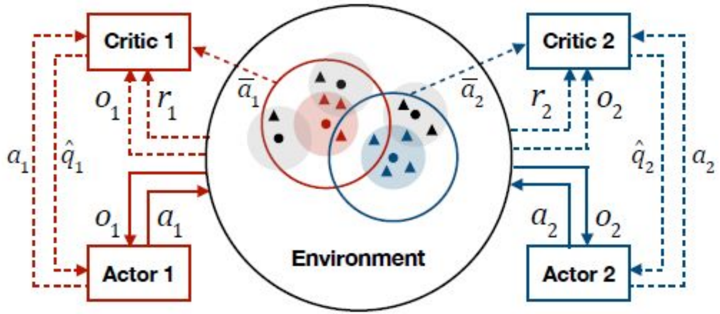


Figure 11: Overview of COD

Cooperative Order Dispatching (COD) with Mean Field (MF) Approximation

- Parameterize the global Q function by pairwise interactions
  - Between an agent and the average response **ā** from a sub-population in the same grid node.
- **Average Response, *ā*$_i$**
  - The number of other agents arriving in the same grid node as agent i, divided by the number of available orders for a given agent ***i***.
  - Ex - **ā = ⅔** in the figure provided
- The *average action-value* function is approximately equivalent to that calculated with the *average response*.

$$Q_i(s,\boldsymbol{a}) = \frac{1}{N_i}\sum_{k \in K_i} Q_i(s,a_i,a_k) \approx Q_i(s,a_i,\bar{a})$$

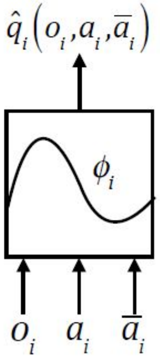


Figure 12: Function approximator with **Average Response, $\bar{a}_i$**

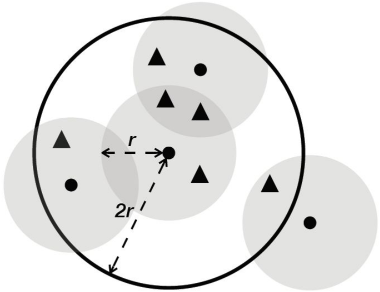


Figure 13: Central Agent with **2 more agents** in the neighborhood and **3 orders.**

Simulator Design

1. Update vehicle status (setting some offline, and bringing some new vehicles online).
2. Provides an observation o^t with a set of active drivers and available orders. Each order contains the origin and destination grid node IDs, while each driver contains the grid node ID of their current location.
3. **The drivers will not move to other grid nodes before taking a new order.**
4. **The COD algorithm generates an optimal list of driver-order pairs a^t with current policy.**
5. After receiving the driver-order pairs the COD algorithm, the simulator returns new observation o^{t+1} and a list of order fees.
6. **Using this observation, o^{t+1} the COD algorithm will calculate the rewards, r^t for each agent.**

Proposed Model 3:
Hybrid Model - Reallocation & Order Dispatching

Model Setup (Work-in-Progress)

- Use the common Environment setup.
- Reallocate agents to high demand grid nodes using Contextual Actor-Critic (cA2C) RL.
- Use Cooperative Order Dispatching (COD) with Mean Field (MF) Approximation to assign agents (drivers) with available orders.
- The model will have
  - 2 separate Critic Networks (Centralized)
  - 2 separate Actor Networks (Individual Agents, N)

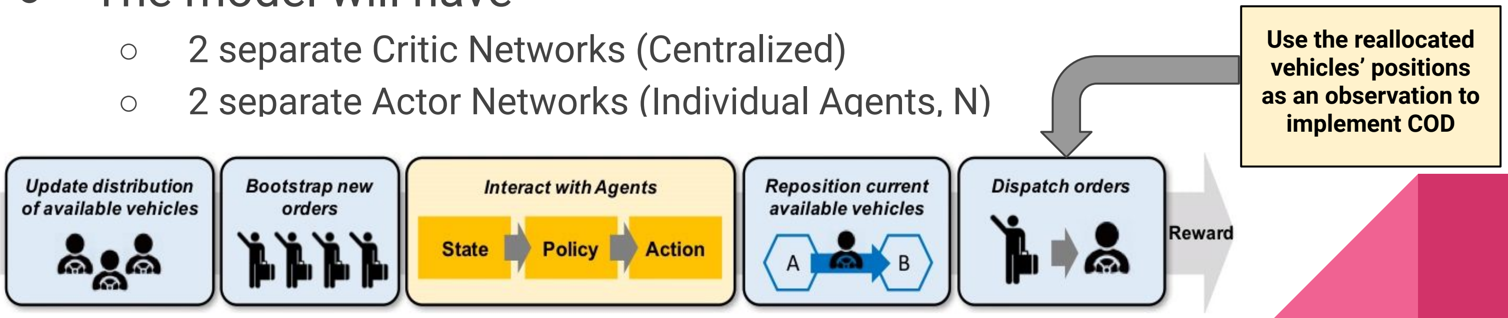
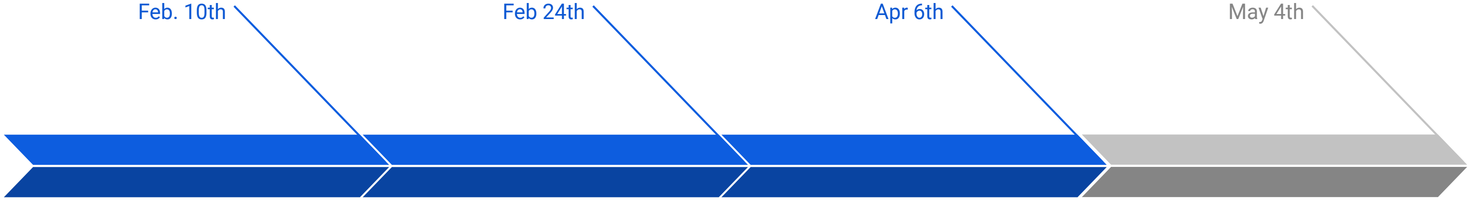


Figure 14: Simulator timeline in one time step (10 min.)

Evaluation Metrics

- Gross Merchandise Volume (GMV)
    - The value of all the orders served in a single Episode (one day = 144 time steps).
- Order Response Rate (ORR)
    - The number of orders taken divided by the number of orders generated.
- Average Order Destination Potential (ADP)
    - The sum of Destination Potential (*DP = #DD - #DS*) of all orders divided by the number of orders taken.

Project Plan



Phase 0

Project proposal

Phase I

Defining the Solution Model
Plans for implementation.

Phase II

First Demo and Plans for completion.
Feedback

Phase III

Final Presentation, Demo and Report Submission.

References

[1] B. Schaller, “The new york city taxicab fact book,” Schaller Consulting, mars, 2006.
[2] NYC Taxi and Limousine Commission, “Taxi of tomorrow survey results,” http://www.nyc.gov/html/tlc/downloads/pdf/tot survey results 02 10 11.pdf, 2011.
[3] Lowe, Ryan, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel and Igor Mordatch. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.” ArXiv abs/1706.02275 (2017): n. Pag.
[4] Lin, Kaixiang et al. “Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning”. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1774-1783. ACM, 2018.
[5] Li, Minne et al. “Efficient Ridesharing Order Dispatching with Mean Field Multi-Agent Reinforcement Learning.” WWW (2019).
[6] H3: Uber’s Hexagonal Hierarchical Spatial Index. https://eng.uber.com/h3/
[7] Artificial Intelligence in Transportation: https://outreach.didichuxing.com/tutorial/kdd2018/
[8] Deep Reinforcement Learning with Applications in Transportation: https://outreach.didichuxing.com/tutorial/kdd2018/
[9] UCL Course on RL: https://www.davidsilver.uk/teaching/

Demo