Scalable Deep Reinforcement Learning
for Ride-Hailing

Mark Gluzman
Center for Applied Mathematics, Cornell University

joint work with
Jiekun Feng (Cornell) and Jim Dai (CUHK-Shenzhen and Cornell)

American Control Conference 2021
Three major companies (Lyft, Uber, and Didi Chuxing) together serve more than 50 million passengers per day\(^1\).

One of the important goals for these companies is to provide a reliable, trustworthy means of transportation, able to fulfill most, if not every, passenger’s request\(^2\).

In our work we focus on optimizing ride-hailing service control policy to maximize the matching rate of passengers and drivers.

\(^1\)M. Schlobach and S. Retzer, “Didi chuxing - How China’s ridehailing leader aims to transform the future of mobility,” in Sustainable Transport in China, 2018

A service territory is divided into $R$ regions.

The service network consists of $N$ cars distributed across the service territory.
Each working day starts at the same time and lasts for $H$ minutes.

No. of new passengers at region $o$ in the $t$-th minute = Poisson($\lambda_o(t)$).
A passenger from region $o$ travels to region $d$ with probability $P_{od}(t)$.

Duration of each trip is deterministic and equals to $\tau_{od}(t)$ (not essential).
Each passenger has a deterministic patience time equals to $L$ minutes.

If the system assigns a car, the passenger and the driver have to accept the matching.
The Centralized Planner:

- In real time, the centralized planner receives ride requests, observes the location and activity of each car in the system.

- Three types of tasks for available cars:
  1. car-passenger matching, 2. empty-car routing, and 3. “do nothing”.

Empty-Car Routing:

- The centralized planner may let the empty car stay at the destination (“do nothing”) or relocate to another region (“empty-car routing”).
Optimal Control Problem Formulation

- Our goal is to find a control policy for the centralized planner that maximizes the total number of car-passenger matchings during one working day by the entire ride-hailing service.

- We formulate a finite-horizon, discrete-time, undiscounted Markov Decision Process (MDP) problem.

- **Challenge #1**: The centralized planner does not know traffic parameters (passengers arrival rates $\lambda_o(t)$, travel times $\tau_{od}(t)$, destination probabilities $P_{od}(t)$).

- **Challenge #2**: Large action space.
Challenge #2: Car-Passenger Matching

- At each epoch, the centralized planner should address all available cars (cars that either idle or are $L$ minutes away from their destinations).
Challenge #2: Empty-Car Routing and “Do Nothing”
Curse of Dimensionality

- Challenge #2: The centralized planner chooses among more than $3^{I_t}$ actions, if there are $I_t$ available cars at decision epoch $t$.
- Recall: three types of tasks for the available cars:
  1. car-passenger matching,
  2. empty-car routing,
  3. “do nothing”.
Challenge #1: Reinforcement Learning, PPO

- A Reinforcement Learning (RL) problem refers to a (model-free) Markov decision process (MDP) problem in which
  - the underlying dynamics is unknown,
  - but optimal actions can be learned from sequences of observed data (states, actions, and rewards).

- We will use Proximal Policy Optimization (PPO)\(^3\) algorithm to optimize the control.

- The PPO method has become a default algorithm for control optimization in new challenging environments including robotics\(^4\), multiplayer video games \(^5\), queueing networks\(^6\).

---


Each action $a_t \in \mathcal{A}$ can be represented as

$$a_t := (a_{t,1}, a_{t,2}, \ldots, a_{t, I_t}),$$

where atomic action $a_{t,i}$ represents a task for one of the available cars.

Idea: sequentially generate atomic actions until each available car is addressed.
Let $s_{t,i}$ be a system state at time $t$, after $i$ atomic actions have been generated.

$$a_{t,i} = \pi(s_{t,i}), \text{ where } \pi : S \rightarrow \mathbb{R}^2 \text{ is a control policy.}$$

Atomic action is generated as $a_{t,i} = (o, d)$, where $o =$origin and $d =$destination:

There are $R^2$ atomic actions, where $R$ is a number of regions.

Which driver should be assigned to action $a_{t,i} = (o, d)$ and what task to fulfill?
Sequential Decision Making Process

If the centralized planner chooses atomic action $a_{t,i} = (o, d)$ then:

- **Which driver?:** an arbitrary driver at region $o$ is assigned to fulfill action $a_{t,i}$.

- **What task?:**
  - if there is a passenger that requested trip $(o, d)$, the driver and passenger are matched.
  - if there is no such a passenger:
    - if $o = d$, the car idles until next decision epoch.
    - if $o \neq d$, the car drives to region $d$ with no passenger (empty-car routing).

As a result, one of the available drivers gets a specific task to fulfill.
Sequential Decision Making Process at epoch $t$

- $a_t = (a_{t,1}, ...)$
Sequential Decision Making Process at epoch $t$

$$a_t = (a_{t,1}, a_{t,2}, \ldots)$$
Sequential Decision Making Process at epoch $t$

$$a_t = (a_{t,1}, a_{t,2}, \ldots a_{t,I_t}),$$

where $I_t$ is a number of available cars just before epoch $t$. 
Assume our current policy is $\pi_\xi$, $\xi \in \Theta$.

We want to find new policy $\pi_\theta$, $\theta \in \Theta$ that outperforms $\pi_\xi$.

The system runs for $K$ working days (episodes) under policy $\pi_\xi$:

$$D_\xi^{(K)} := \left( \begin{array}{c}
\text{day 1, for each } t: (s_{t,1}, a_{t,1}), (s_{t,2}, a_{t,2}), \\
\text{day 2, for each } t: (s_{t,1}, a_{t,1}), (s_{t,2}, a_{t,2}), \\
\ldots, \\
\text{day } K, \text{ for each } t: (s_{t,1}, a_{t,1}), (s_{t,2}, a_{t,2}) \end{array} \right).$$

Experiments: 9-region network

- The 9-region transportation networks from Braverman et al. 2019 is based on the real-world data released by the Didi Research Institute.

![Graph showing percentage of fulfilled ride requests over policy iterations]

A transportation network consisting of $R = 9$ regions, $N = 2000$ cars, and $H = 240$ minutes.

“Time-dependent lookahead” policy (requires model knowledge) from Braverman et al. 2019.

---