

Efficient Quasi-optimal Solution for Ride Sharing Task

– An automatic planning approach in constrained combination optimization

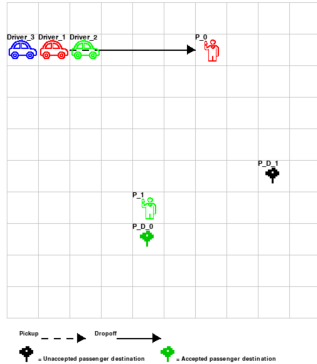
Ziyue Chu, Leonardo Stella from DLC Group

School of Computer Science
University of Birmingham

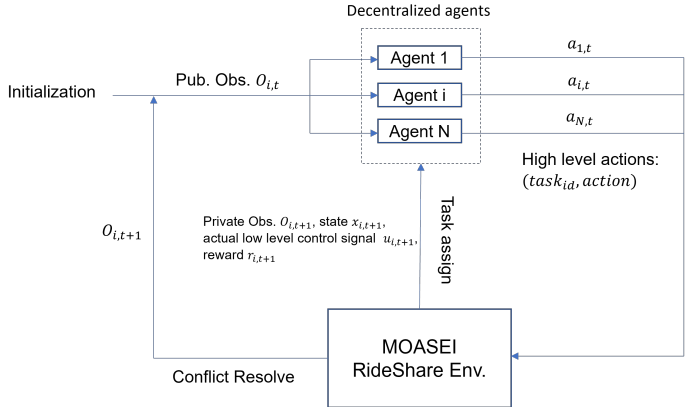
AAMAS'26 Competition: MOASEI 2026 May 25

- ① Environment Setting
- ② Baseline
- ③ Our Methods
- ④ Results
- ⑤ Future Work & Open Questions

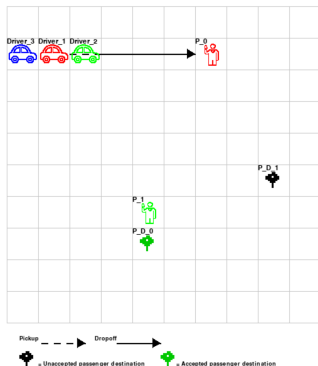
Ride Sharing Task:



Decentralized Multi-agent Control Process in Ride Sharing Task



Ride Sharing Task as Parallel POMDPs:



Each agent $i \in [M]$:

- Obs. Space: Pub. + Pri. Obs.:
 $(y_0, x_0, y_d, x_d, task_status, fare)$
 - Pub. Obs.: All available tasks in the system at t
 - Pri. Obs.: Tasks already taken by this agent
- State Space: $(y_t, x_t, num_{accept}, num_{riding})$
- Action Space: $(task_{id}, action)$
 - $action = \{1, 2, 3\}$: accept, to pickup, to dropoff
 - High Level action $\mapsto$ actual control $\{\uparrow, \downarrow, \leftarrow, \rightarrow, -\}$
 - Variant Obs. space $\rightarrow$ Open tasks
- Rewards: -0.8/step, fares for finished tasks (sparse)

Greedy Algorithms:

- Most widely applied algo. in industry
- At time t , always do the **best** task: accept, pickup, dropoff sequentially
 - Nearest pickup position
 - Most fare gain / travel dist.
 - Most fare gain / total dist. (inc. to pickup)
- Until the prev. task is finished, find next greedy task

Key Performance Observations:

1. Take only 1 passenger a time
 2. Waste many steps on dispatching
 3. No planning for better future tasks
 4. Negative episodic rewards
- ⇒ Inefficiency → non-profitable
(fatal in learning, ineffective data)

Combination Optimization:

- Init.: Each agent stores a scheduled traj. $\mathcal{L}_{i,t}$ with finished task popped (the first element of $\mathcal{L}_{i,t}$ always points to the current task)
- At time t , each agent $i \in [M]$ do traj. planning from its tasks observation:
 - Constrained Vehicle Routing Problem (CVRP)
 - i. Pickup Before Dropoff
 - ii. Agent Capacity: max 4 accepted passenger tasks: $||\mathcal{L}_{i,t}|| \leq 8$
 - iii. Add max $4 - \sum_j \mathbf{1}[task_j \in \mathcal{L}_{i,t}]$ tasks from public observations
 - Instead of calculating all possible combinations, we only update $\mathcal{L}_{i,t}$ with one profitable traj. $\mathcal{L}_{i,t} = (task_j)$, s.t. $\sum_j fare_j \geq \text{dist}(\mathcal{L}_{i,t}) * 0.8^1$
- If $task_j \in \mathcal{L}_{i,t}$ is in public obs. $\rightarrow action = [j, 0] \rightarrow END$ (accept new tasks first)
- o.w. Execute scheduled traj., check $\mathcal{L}_{i,t}[0] \in task_j$ status:
 - If $task_j$ not on ride $\rightarrow action = [j, 1] \rightarrow END$ (to pickup);
 - If $task_j$ on ride $\rightarrow action = [j, 2] \rightarrow END$ (to dropoff);
 - If $task_j$ not in obs. list $\rightarrow$ Task finished $\rightarrow \mathcal{L}_{i,t}.pop(task_j)$

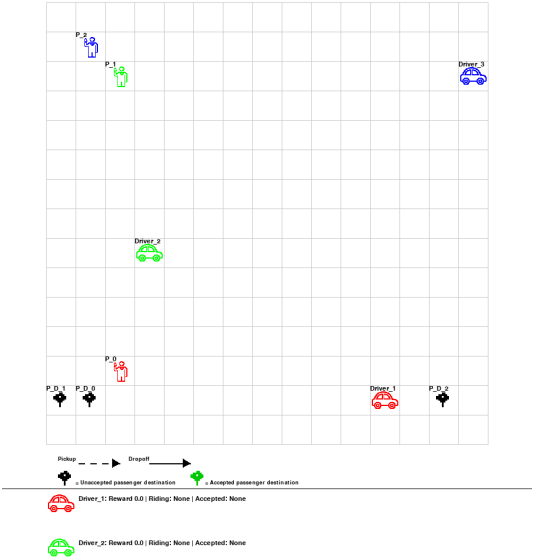
¹Coefficient 0.8 is from Env. It can also be a variant hyperparam. to balance quality and efficiency

Envs/Algos.	ridesharing0	ridesharing1	ridesharing2
Ours	-0.2 ± 4.8	53.788 ± 13.082	89.463 ± 4.55
Greedy	-83.387 ± 4.224	-91.112 ± 9.288	-5351.433 ± 2482.273^2
Random	-23130.234 ± 515.948	-23002.035 ± 308.977	-24331.707 ± 380.76

Table: Performance (system total rewards) comparison across 3 environments in 100 steps

²These experiments accepts more than capacity (4) passengers and receive -100 punishment. 

Time: 1



From planning to learning:

- Task-based action space: No predictions ahead of observed tasks (e.g. idle agents can move to CBD in advance for peak hours given history data)
- Not supported in current simulation environment
- Goal: Learn the same quasi-optima traj. efficiently given all available tasks
- Challenge: learn valid action sequential constraints (especially in learning phase)

Open Questions:

- Conflict resolve: e.g. Agents bid for the same tasks in their traj. Assign to which?
- Coop. or Compete? System Mechanism design: VCG auction? Coalition form?
- Efficient computation for a profitable traj. in super large scale systems (current complexity $\Omega(N^K))^3$

³N: Num of available tasks; K: Agent capacity, constant number, here is 4.

Thank you! Questions?

Contributor

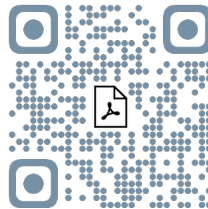
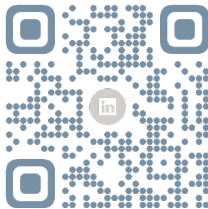
Ziyue Chu*

MARL Researcher from DLC Research Group of UoBirmingham

zero-cooper.github.io

Research Directions:

- Networked-MARL Thoery
- Learning Cooperative Games
- UAV Swarms Navigation



zxc332@student.bham.ac.uk



[Zero-Cooper_work](https://github.com/Zero-Cooper/Zero-Cooper)