Stanford | ENGINEERING
Computer Science

Leveraging Multi-Agent Reinforcement Learning to Enhance E-AMoD Fleet Control

Ayush Agrawal, Aaryan Singhal, Justin Wu {ayushag, aaryan04, justwu} @ cs.stanford.edu

Summary

- Electric-driven AMoD systems are critical for growing urban transportation demands, but it's difficult to manage asymmetry, profit, and charging needs city-wide.
- Previous works ignore charge as a key variable & only use linear or single-agent methods.
- Divided problem into a tri-level framework, with GNN-represented AMoD environments.
- Developed a novel sequential multi-agent coordination mechanism with dual focus of maximizing profit & recharging effectively.
- One-time freeze method performs best: +37% improvement over standard MARL.
- Had to use solvable toy problem: one-time freeze still achieves 91% of linear optimizer.

Relevant Background

Enabling AMoD Systems:

- Increasing demand for urban transit: addressed by autonomous EVs (fast, green, flexible).
- Asymmetric demand, fast control, profit generation, & charging needs must be optimized.

Initial Approaches at Maximizing Profits:

- Linear MPC meant for small-scale scenarios, fails with scaling to city-wide systems.
- Early RL approaches with DQNs & PPO only optimized one vehicle at a time: $O(n^n)$.

Graph Neural Networks, Tri-level Framework, & Charge:

- `Gammelli et al.` use GNNs to model inter-vehicle coordination in one timestep [1].
- Formulated a tri-level framework for profit maximization:
  1. Assign subset of all vehicles for current passenger demand.
  2. Calculate a desired distribution of idle vehicles to serve future timesteps to maximize revenue.
  3. Rebalance vehicles accordingly, minimizing cost.
- Accounting for charge & energy prices hard to solve linearly: requires MARL.

Traditional MARL Strategies are Ineffective:

- Must cooperatively combine *sequential* (i.e. joint) policies of *heterogeneous agents*.
- Difficult to reach convergence & stable performance in multi-agent system.
- CTDE attempts to generalize a global joint-value function with locally independent agents: ineffective since non-constant global state and non-homogeneous agents [2].
- HATRPO/HAPPO: developed for heterogeneous agents, but both still require explicit end objective (which cannot be heuristically determined in our optimization problem).
- Must allow agents to:
  1. Intrinsically understand their sequential nature.
  2. Explicitly communicate between themselves and advise each other on optimal actions.
  3. Prevent overfitting and mutual degradation of performance due to sub-optimal local policies.

Acknowledgements & References

We'd like to thank Stanford's Autonomous Systems Lab for inspiring this research.

[1] Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco C. Pereira, and Marco Pavone. Graph neural network reinforcement learning for autonomous mobility-on-demand systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, page 2996–3003. IEEE Press, 2021.

[2] Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-agent conditional policy factorization. In *The Eleventh International Conference on Learning Representations*, 2023.

Methods + Experiments

Environment:

- The Linear Optimizer easily solves dispatching in the first toy problem. Must make problem more complex in order to observe true RL comparisons.
- Nonlinearly varied the demand and charge over time, increased number of spatial nodes, and significantly increased the number of vehicles relative to passenger demand.

Baselines:

- Model Predictive Control (MPC) approach based on network flow models provides a theoretical optimum for reward, demand, and rebalancing.
- Use linear optimizer for dispatching + charge allocation and RL for rebalancing.

Framework:

- Use an Advantage Actor-Critic algorithm with GNN parametrization for Actor and Critic.
- We use linear optimization or a GNN Actor-Critic RL Agent in order to dispatch idle vehicles to specific trip requests.
- In rebalancing, node features (charge, cost, etc) are updated with graph convolutions, and then passed through non-linearities to compute parameters α of the Dirichlet policy (probability distribution over stations, indicating the percentage of idle vehicles to be rebalanced in each station) and an estimate of the value function $V(s_t)$.
- The total reward combines the dispatching reward and the rebalancing cost

$$Reward = \sum_{i,j} S_{ij}(P_{ij}(T_{ij} + T_{norm})C_{oper}) - \sum_{i,j} R_{ij}((T_{ij} + T_{norm})C_{oper} + P_{avg}Q_{ij})$$

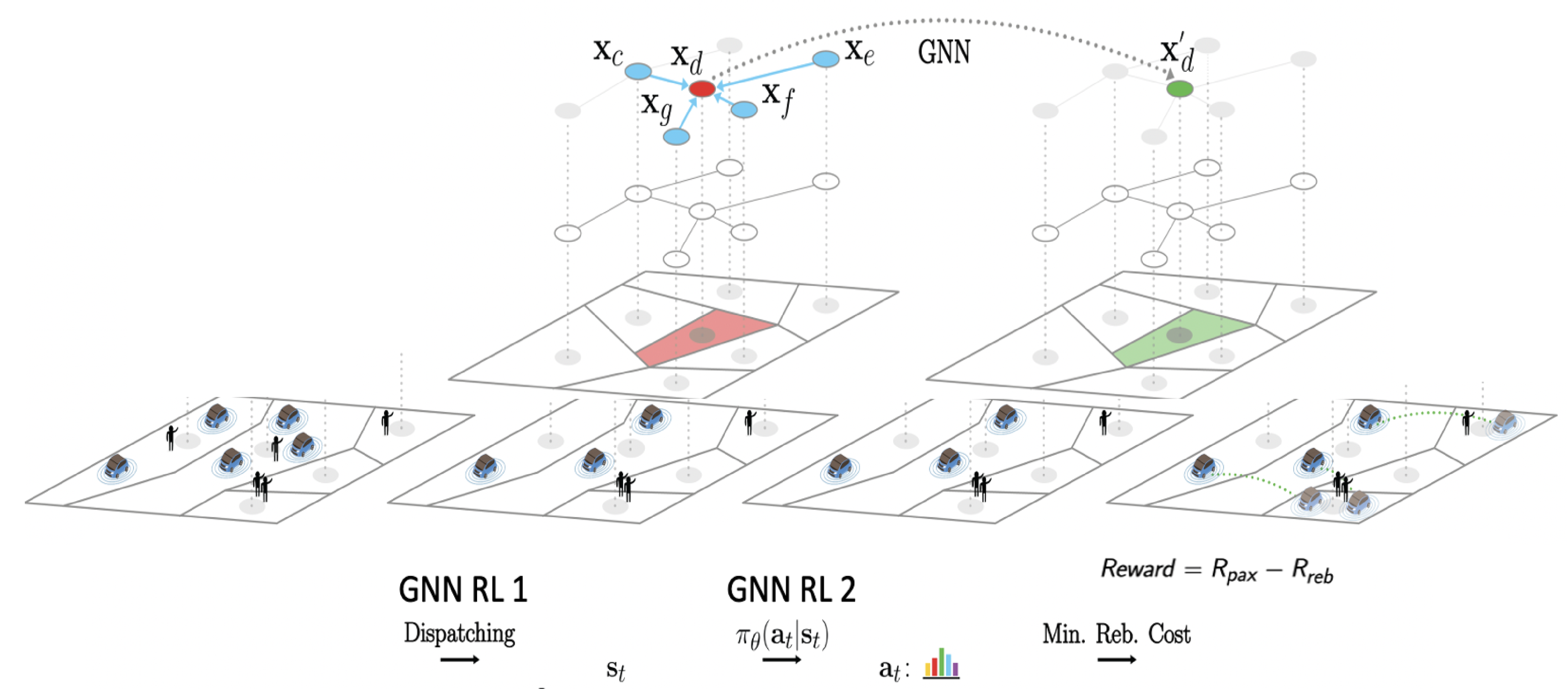

Figure 1. For each time-step, a tri-level framework is utilized to (1) assign passengers to vehicles, (2) rebalance idle vehicles across the space-charge graph, and (3) optimally achieve the desired rebalancing of vehicles

Experiments:

1. **Baseline:** Linear optimization for dispatching, RL2 for rebalancing.
2. **Standard MARL:** Multiagent RL1 and RL2 with no freezing.
3. **N-alternate freezing:** Switch freezing of RL1 & RL2 every $n = 2000$ episodes.
4. **Warm-start MARL:** Baseline for 8k episodes (convergence), then standard MARL.
5. **One-time freeze:** Let RL2 converge on a linear optimizer for 8k episodes, then freeze RL2 and let RL1 train for remaining 8k episodes.

Results

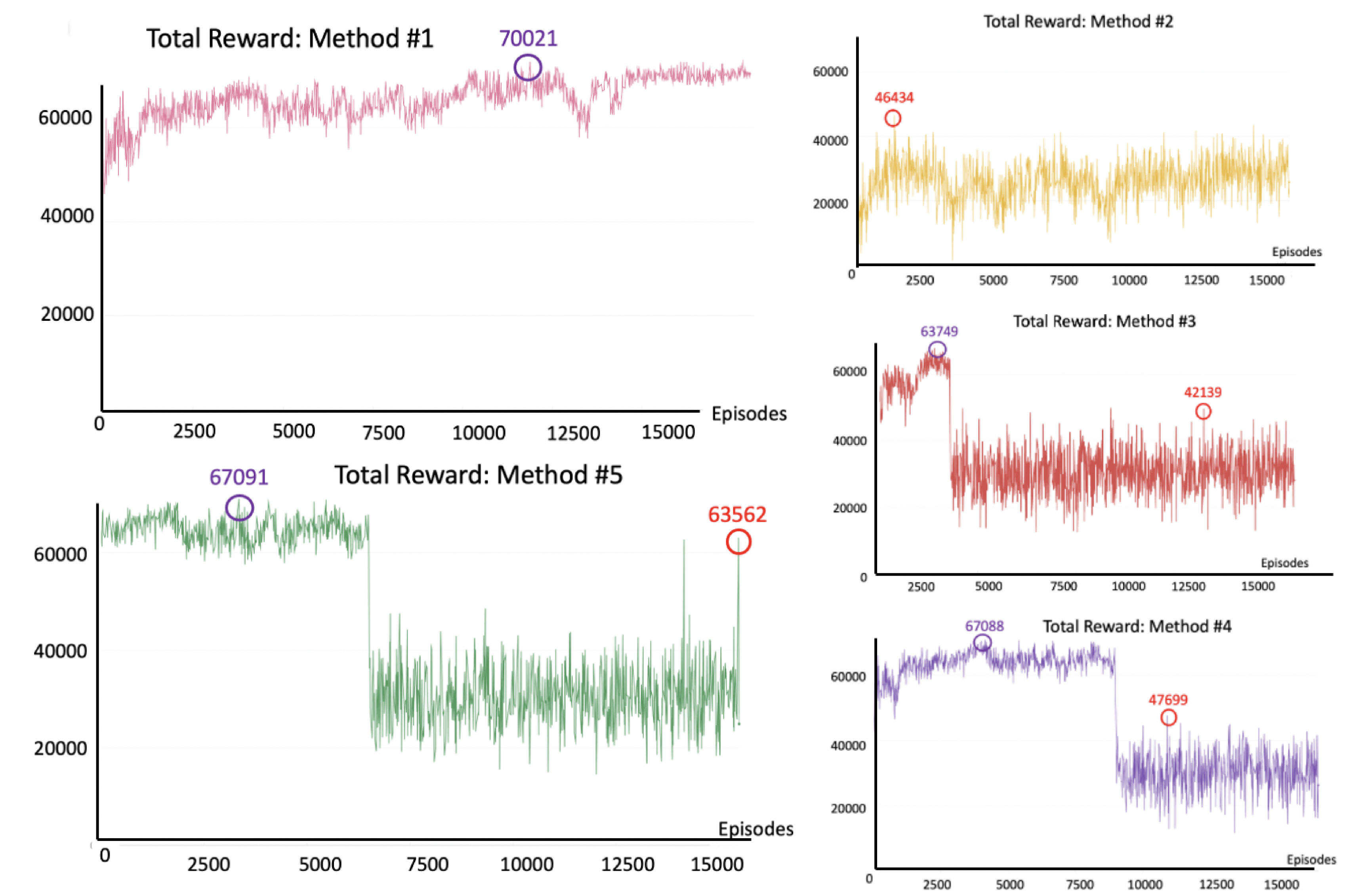

Figure 2. Comparison of best reward achieved by different configurations of the tri-level framework

Experiment	Best Reward	Served Demand	Rebalance Cost
Baseline	70021	8965	12597
Standard MARL	46434	11594	10731
N-alternate freezing	42139	13221	8136
Warm-start MARL	47699	12585	10638
One-time freeze	63562	11205	12134

Table 1. Breakdown of best reward, highest served demand, and optimal rebalancing cost for different tri-level configurations

Results & Analysis

- MPC determines theoretically optimal answers (tries every possible combination): **Reward = 77.7k**, Served Demand = 9.2k, Rebalance Costs = 9.6k.
- Exp. 1 (baseline) converges to 71k reward: updated toy problem still **not hard enough**.
- **Exp. 5 performs best** out of MARL strategies, outperforming each by > 33%.
- **Moving targets problem** within MARL causes mutual degradation of both agents' performance: only addressed by Exp. 5.
- None of the MARL strategies converge, as expected. Exp. 5 achieved a new optimum every 2,000 episodes: perhaps longer training would have allowed even better performance **(time and compute constraints)**.
- Optimal solution is problematic: incentivizes serving less customers to maximize profit.
- **Hyperparameter Tuning:** Achieved best w/ GCN RL1 + MPNN RL2, no self-loops.
- **Future work**: apply to real world-scale datasets (NYC/SF), experiment with CADP & MARL-Transformer approaches (both released very recently).