Denne artikel er udgivet i det elektroniske tidsskrift
Artikler fra Trafikdage på Aalborg Universitet
(Proceedings from the Annual Transport Conference at Aalborg University)
ISSN 1603-9696
https://journals.aau.dk/index.php/td

trafikdage
NY VIDEN & NETVÆRK

Reinforcement Learning for Autonomous Mobility-on-Demand Systems

Daniele Gammelli, daga@dtu.dk
Department of Technology Management and Economics, Technical University of Denmark

Kaidi Yang, kaidi.yang@nus.edu.sg
Department of Civil and Environmental Engineering, National University of Singapore

James Harrison, jamesharrison@google.com
Google Research, Brain Team

Filipe Rodrigues, rodr@dtu.dk
Department of Technology Management and Economics, Technical University of Denmark

Francisco C. Pereira, camara@dtu.dk
Department of Technology Management and Economics, Technical University of Denmark

Marco Pavone, *pavone@stanford.edu*
Department of Aeronautics and Astronautics, Stanford University

Abstract

Autonomous mobility-on-demand (AMoD) systems represent a rapidly developing mode of transportation wherein travel requests are dynamically handled by a coordinated fleet of robotic, self-driving vehicles. Given a graph representation of the transportation network - one where, for example, nodes represent areas of the city, and edges the connectivity between them - we argue that the AMoD control problem is naturally cast as a node-wise decision-making problem. In this paper, we propose a deep reinforcement learning framework[1] to control the rebalancing of AMoD systems through graph neural networks. Crucially, we demonstrate that graph neural networks enable reinforcement learning agents to recover behavior policies that are significantly more transferable, generalizable, and scalable than policies learned through other approaches. Empirically, we show how the learned policies exhibit promising zero-shot transfer capabilities when faced with critical portability tasks such as inter-city generalization, service area expansion, and adaptation to potentially complex urban topologies.

[1] Code available at: https://github.com/DanieleGammelli/gnn-rl-for-amod

Introduction

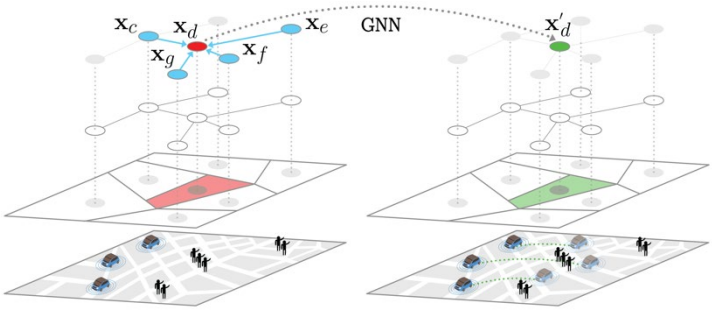

Figure 1. This paper proposes a framework to control AMoD systems by learning a shared rebalancing rule across all areas (nodes) in the transportation network. Through the use of graph convolutions, the proposed architecture aggregates information coming from both local information (in red), as well as information about neighboring areas (in blue), to learn an updated representation (in green) for downstream rebalancing tasks.

Personal urban mobility is currently dominated by the increase of private cars for fast and anytime point-to-point travel within cities. However, this paradigm is currently challenged by a variety of impellent factors, such as the production of greenhouse gases, dependency on oil, and traffic congestion, especially in densely populated areas. With the urban population projected to reach 60 percent of the world population by 2030 [1], private cars are widely recognized as unsustainable for the future of personal urban mobility. In light of this, cities face the challenge of devising services and infrastructure that can sustainably match the growing mobility needs and reduce environmental harm.

In order to address this problem, any potential solution will likely need to work towards the convergence of a variety of emerging technologies [2]. To this regard, one of the most promising strategies is the concept of *mobility-on-demand* (MoD), in which customers typically request a one-way ride from their origin to a destination and are served by a shared vehicle belonging to a larger fleet. One of the major limitations of the MoD paradigm lies in the spatio-temporal nature of urban mobility, such that trip origins and destinations are asymmetrically distributed (e.g., commuting into a downtown in the morning and vice-versa in the evening), making the overall system *imbalanced* and sensitive to disturbances. Related to this problem, the advancement in autonomous driving technologies offers a potential solution. Specifically, autonomous driving could enable an MoD operator to coordinate vehicles in an automated and centralized manner, thus eliminating the need for manual intervention from a human driver. However, controlling AMoD systems potentially entails the routing of thousands of robotic vehicles within complex transportation networks, thus effectively making the AMoD control problem an open challenge.

In this work, we propose the use of graph neural networks to centrally control AMoD systems. In particular, given a graph representation of the transportation network - a graph where nodes represent areas of the city and edges the connectivity between them [3] - we learn a node-wise rebalancing policy through deep reinforcement learning. We argue that graph neural networks exhibit a number of desirable properties and propose an actor-critic formulation as a general approach to learn proactive, scalable, and transferable rebalancing policies.

Methodology

In this section, we propose a control framework to learn effective AMoD rebalancing policies from experience. Towards this aim, we consider a transportation network represented by a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with M single-occupancy vehicles, where $\mathcal{V}$ represents the set of stations (e.g., pick-up or drop-off locations) and $\mathcal{E}$ represents the shortest paths connecting the stations. Let us denote $N_v = |\mathcal{V}|$ as the number of stations. The time horizon is discretized into a set of discrete intervals $\mathcal{T} = \{1,2,\cdots,T\}$ of a given length ΔT. The travel time for edge $(i,j) \in \mathcal{E}$ is defined as the number of time steps it takes a vehicle to travel along the shortest path between station i and station j, denoted as an integer $\tau_{ij} \in \mathbb{Z}_+$. Further denote c_{ij} as the cost of traveling through an edge $(i,j) \in \mathcal{E}$, which can be calculated as a function of travel time τ_{ij}. At each time step, customers arrive at their origin stations and wait for vehicles to transport them to their desired destinations. The trip traveling from station $i \in \mathcal{V}$ to station $j \in \mathcal{V}$ at time step t is characterized by demand d_{ij}^t and price p_{ij}^t. Consequently, passengers departing from origin station i at

time t will arrive at the destination station j at time $t + \tau_{ij}$. The AMoD operator coordinates a fleet of taxi-like fully-autonomous vehicles to serve the transportation demand. The operator matches passengers to vehicles, and the matched vehicles will deliver passengers to their destinations. Let us denote $x_{ij}^t \leq d_{ij}^t$ as the passenger flow, i.e. the number of passengers traveling from station i to station j at time step t that are successfully matched with a vehicle. Passengers not matched with any vehicles will leave the system. For vehicles not matched with any passengers, the operator will either have them stay at the same station or rebalance them to other stations. Let us denote y_{ij}^t as the rebalancing flow, i.e., the number of vehicles rebalancing from station i to station j at time step t.

A Three-Step Framework. We adopt a three-step decision-making framework similar to [?] to control an AMoD fleet: (1) deriving passenger flow by solving a matching problem, (2) computing the desired distribution of idle vehicles at the current time step by using the learned policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$, (3) converting the desired distribution to rebalancing flow by solving a minimal rebalancing-cost problem. Notice that in the three-step procedure, we have an action at each node as opposed to along each edge (as in the majority of literature). In other words, we reduce the dimension of the action space of the AMoD rebalancing MDP to N_v (compared to N_v^2 in edge-based approaches), and thus significantly improve scalability of training and implementation.

We now explain in more detail the three-step decision framework that the AMoD operator employs at each time step t. The first step is passenger matching, wherein the following matching problem is solved to derive passenger flows $\{x_{ij}^t\}_{i,j\in\mathcal{V}}$:

$$\max_{\{x_{ij}^t\}_{i,j\in\mathcal{V}}} \quad \sum_{i,j\in\mathcal{V}} x_{ij}^t\left(p_{ij}^t - c_{ij}^t\right) \qquad (1a)$$

$$\text{s.t.} \quad 0 \leq x_{ij}^t \leq d_{ij}^t, i,j \in \mathcal{V}, \qquad (1b)$$

where the objective function Eq.(1a) represents the total profit of passenger assignment calculated as the difference between revenue and cost, and the constraint Eq.(1b) ensures that the passenger flow is non-negative and does not exceed the demand. Notice that since the constraint matrix is totally unimodular, the resulting passenger flows are positive integers, i.e., $x_{ij}^t \in \mathbb{Z}_+$if $d_{ij}^t \in \mathbb{Z}_+, \forall i,j \in \mathcal{V}$.

The second step entails determining the desired idle vehicle distribution $\mathbf{a}_{\text{reb}}^t = \{\mathbf{a}_{\text{reb},i}^t\}_{i\in\mathcal{V}}$, where $\mathbf{a}_{\text{reb},i}^t \in [0,1]$ defines the percentage of currently idle vehicles to be rebalanced towards station i in time step t, and $\sum_{i\in\mathcal{V}} \mathbf{a}_{\text{reb},i}^t = 1$. With desired distribution $\mathbf{a}_{\text{reb}}^t$, denote $\hat{m}_i^t = \left\lfloor a_{reb,i}^t \sum_{i\in\mathcal{V}} m_i^t \right\rfloor$ as the number of desired vehicles, where m_i^t represents the actual number of idle vehicles in region i at time step t. Here, the floor function $\lfloor\cdot\rfloor$ is used to ensure that the desired number of vehicles is integer and always available $\left(\sum_{i\in\mathcal{V}} \hat{m}_i^t \leq \sum_{i\in\mathcal{V}} m_i^t\right)$. Reinforcement learning will thus be used to choose meaningful desired distributions $\mathbf{a}_{\text{reb}}^t$ through a learned policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$

The third step entails rebalancing, wherein a minimal rebalancing-cost problem is solved to derive rebalancing flows $\{y_{ij}^t\}_{(i,j)\in\mathcal{E}}$:

$$\min_{\{y_{ij}^t\}_{(i,j)\in\mathcal{E}} \in \mathbb{Z}_+^{|\mathcal{E}|}} \sum_{(i,j)\in\mathcal{E}} c_{ij} y_{ij}^t \qquad (2a)$$

$$\text{s.t.} \sum_{j\neq i} \left(y_{ji}^t - y_{ij}^t\right) + m_i^t \geq \hat{m}_i^t, i \in \mathcal{V}, \qquad (2b)$$

$$\sum_{j\neq i} y_{ij}^t \leq m_i^t, i \in \mathcal{V}, \qquad (2c)$$

Table 1: System Performance on Chengdu 4×4 Network

	Reward (%Dev. MPC)	Served Demand	Rebal. Cost ($)
ED	12,538 (-19.2%)	41,189	3,397
CQL	12,334 (-20.5%)	41,273	4,174
A2C-MLP	12,530 (-19.2%)	41,076	3,899
A2C-CNN	13,274 (-14.5%)	40,904	3,087
A2C-GNN (ours)	15,167 (**-2.2%**)	40,578	1,063
MPC-tri-level	15,516 (0%)	42,425	1,453
MPC-standard	16,702 (7.6%)	44,662	1,162
A2C-GNN-0Shot	14,791 (**-4.7%**)	40,646	1,467

Table 2: System Performance on New York 4×4 Network

	Reward (%Dev. MPC)	Served Demand	Rebal. Cost ($)
ED	30,746 (-10.7%)	8,770	7,990
CQL	30,496 (-11.4%)	8,736	8,284
A2C-MLP	30,664 (-10.9%)	8,773	7,920
A2C-CNN	30,443 (-11.5%)	8,904	8,775
A2C-GNN (ours)	33,886 (**-1.6%**)	8,772	5,038
MPC-tri-level	34,416 (0%)	8,865	4,647
MPC-standard	35,356 (2.7%)	8,968	4,296
A2C-GNN-0Shot	33,397 (**-3.0%**)	8,628	4,743

where the objective function Eq.(2a) represents the rebalancing cost, constraint Eq.(2b) ensures that the resulting number of vehicles (the left-hand size) is close to the desired number of vehicles (the right-hand side), and Eq. (2c) limits the rebalancing flows from a region to the vehicles available in that region.

Experiments and Discussion

In this section, we present simulation results that demonstrate the performance of our proposed approach. 1 Specifically, the goal of our experiments is to answer the following questions: (1) Can the proposed A2C-GNN learn effective rebalancing strategies on real-world urban mobility scenarios? (2) What are the generalization capabilities of a behavior policy learned through our approach? (3) Computationally, what are the advantages of GNN-based RL approaches compared to traditional control-based strategies?

These rebalancing algorithms are evaluated using two case studies inspired by New York City, USA, and the city of Chengdu, China, whereby we study a hypothetical deployment of AMoD systems to serve the morning commute demand in popular areas of Manhattan (8 a.m. -10 a.m., with a size of $4\ \mathrm{km} \times 4\ \mathrm{km}$) and Chengdu (7 a.m. -10 a.m., with a size of $10\ \mathrm{km} \times 10\ \mathrm{km}$), respectively. In our experiments, we care about a set of Key Performance Indicators not included within the reward function. Specifically, we also monitor (i) Served demand: defined as the total number of trips satisfied by the AMoD control strategy, (ii) Rebalancing cost: defined as the overall cost induced on the system by the rebalancing policy, and (iii) *Percentage deviation* from an oracle MPC (Model Predictive Control) performance, having oracle information of future system states (%Dev. MPC).

Chengdu and New York Cases. Results in Tables 1 and 2 show that A2C-GNN can learn rebalancing policies able to achieve close-to-optimal system performance on both tasks. Specifically, A2C-GNN's performance is only 2.2% (Chengdu) and 1.6% (New York) away from the oracle performance. Interestingly, A2C-GNN is able to exploit its learned shared local filter to achieve more than 65% (Chengdu) and 36% (New York) cost savings in its rebalancing trips when compared to learning-based approaches based on different neural architectures. Moreover, by monitoring the number of customers served, we notice how A2C-GNN learns rebalancing policies able to proactively select more profitable trips. Specifically, in Table 1, results show that A 2C-GNN is able to achieve a 14% increase in profit (i.e., reward) compared to the second-best non-oracle approach (A2C-CNN), despite having the lowest number of customers served across all methods.

Computational analysis. We study the computational cost of A2C-GNN compared to MPC-based solutions. As shown in Fig. 2 (right), we compare the time necessary for both approaches to compute a single rebalancing decision. Specifically, we do so across varying dimensions of the underlying transportation network, ranging from 16 up until 400 stations. The results show that, once trained, learning-based approaches allow for fast computation of rebalancing policies by forward-propagation of the current system state through the learned policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$. Most importantly, A2C-GNN exhibits computational complexity linear in the number of nodes and graph connectivity, as opposed to control-based approaches which scale super-linearly in the number of edges [5].

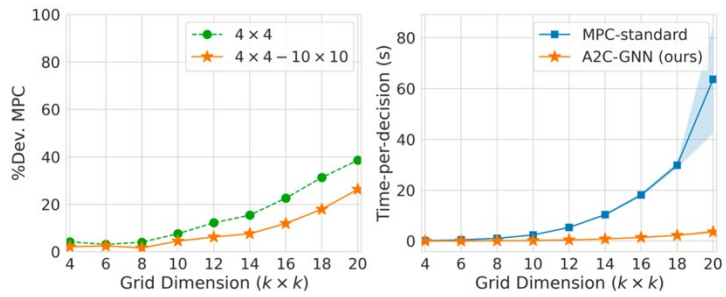

Figure 2: Left: System performance (Percentage Deviation policy, and (iii) Percentage deviation from an oracle MPC from MPC-standard) for agents trained either on a single (Model Predictive Control) performance, having oracle granularity (4×4) or across granularities (4×4-10×10), information of future system states (%Dev. MPC). Right: Comparison of computation times between A2CGNN and MPC-standard.

This work addresses the problem of recovering effective rebalancing strategies for AMoD systems by proposing graph neural networks as a general approach to parametrize policy-based reinforcement learning agents. We introduce an actor-critic algorithm where both policy and value function estimator make use of graph convolutions to define an agent capable of dealing with structured transportation networks. Our experiments focus on real-world case studies and show how the proposed architecture is able to achieve close-to-optimal performance on a variety of scenarios. Crucially, we show how the relational inductive biases introduced by graph neural networks allow reinforcement learning agents to recover highly flexible, generalizable and scalable behavior policies. In future work, we plan to investigate increasing the complexity and stochasticity in the system dynamics, such as considering a mixed fleet of autonomous and human driven vehicles. Given their ability to learn about the system dynamics through interaction with an environment, we believe reinforcement learning approaches to be well suited for challenging, stochastic environments as the ones described by complex human-robot interactions.

References

[1] United Nations. World urbanization prospects: The 2014 revision. Technical report, United Nations, 2014.

[2] M. Pavone. Autonomous Mobility-on-Demand systems for future urban mobility. In Autonomes Fahren. Springer, 2015.

[3] R. Zhang and M. Pavone. Control of robotic Mobility-on-Demand systems: A queueing-theoretical perspective. Int. Journal of Robotics Research, 35(1–3):186–203, 2016.

[4] C. Fluri, C. Ruch, J. Zilly, J. Hakenberg, and E. Frazzoli. Learning to operate a fleet of cars. In Proc. IEEE Int. Conf. on Intelligent Transportation Systems, 2019.

[5] J. van den Brand. A deterministic linear program solver in current matrix multiplication time. In ACM-SIAM Symp. on Discrete Algorithms, 2020.