Graph Neural Network Reinforcement Learning for Autonomous Mobility-on-Demand Systems

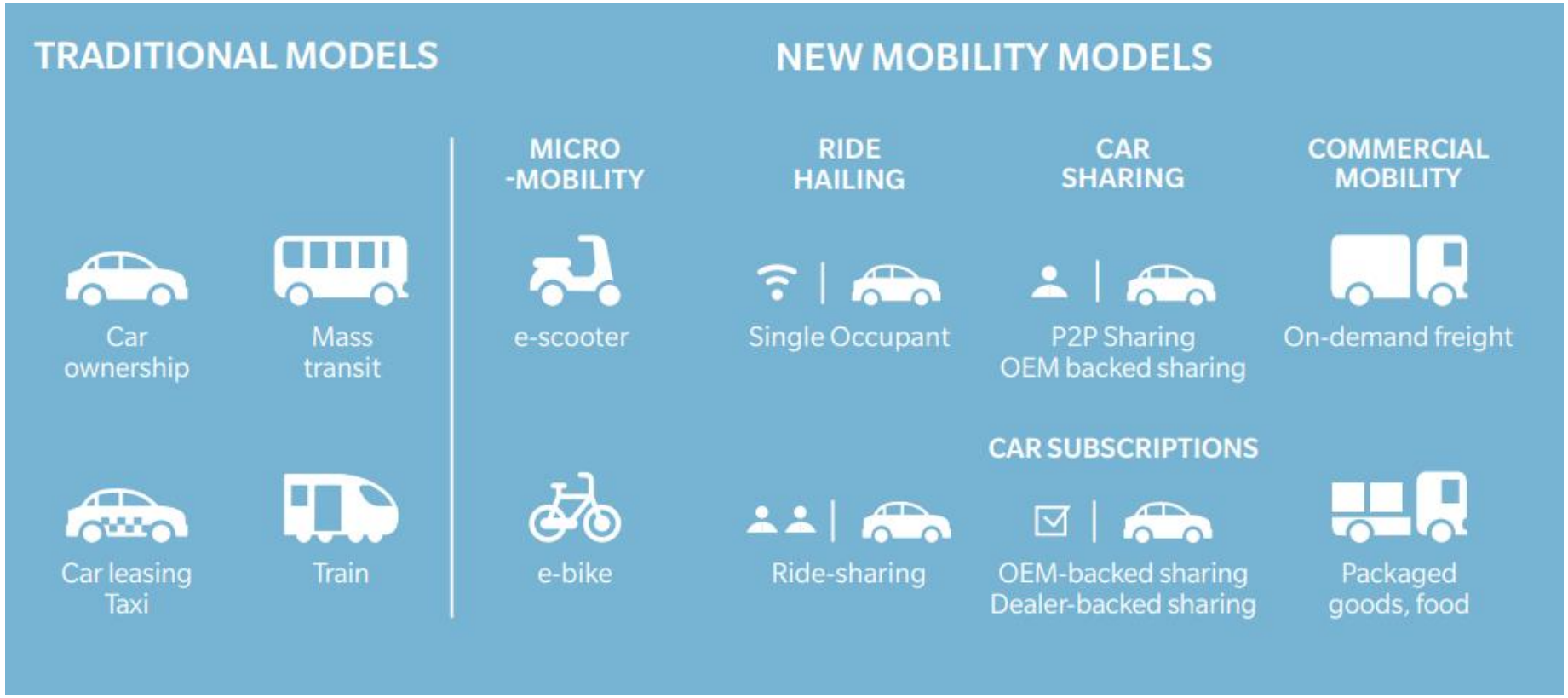
Daniele Gammelli, Kaidi Yang, James Harrison,
Filipe Rodrigues, Francisco C. Pereira, Marco Pavone

발표자: 여지호

Brief introduction of Mobility-on-demand system

Mobility-on-Demand (MoD) System?

- **MaaS (Mobility-as-a-Service)**
  - Traditional transportation modes + New Mobility Models

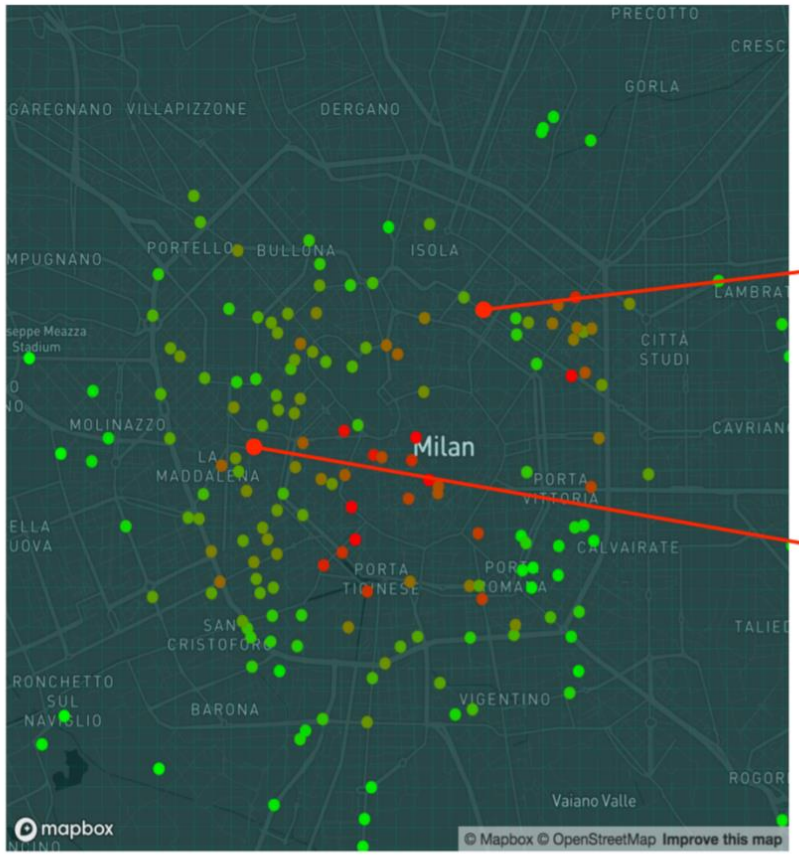


Research topic in AMoD system

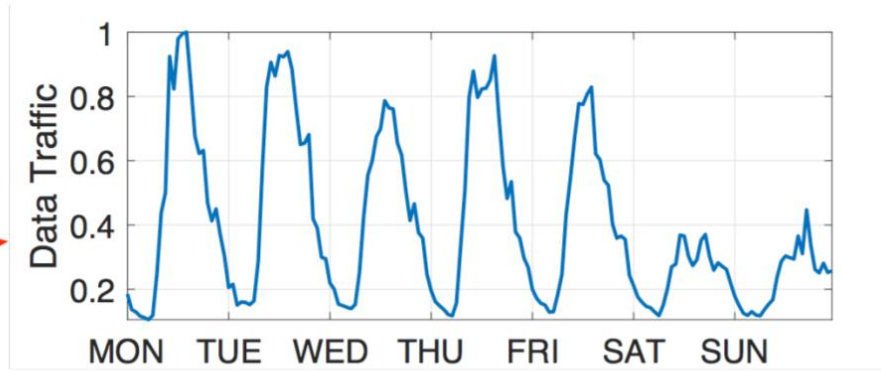
- **Demand Prediction**
- **Dispatch**
  - STDN has a large model for the real environment (Local CNN + LSTM + GCN)
- **Relocation (Rebalancing, Repositioning)**
  - STDN has a large model for the real environment (Local CNN + LSTM + GCN)
  - For the realistic application, need to make a compact model
- **Construct Environment (Digital twin)**

Research topic in AMoD system

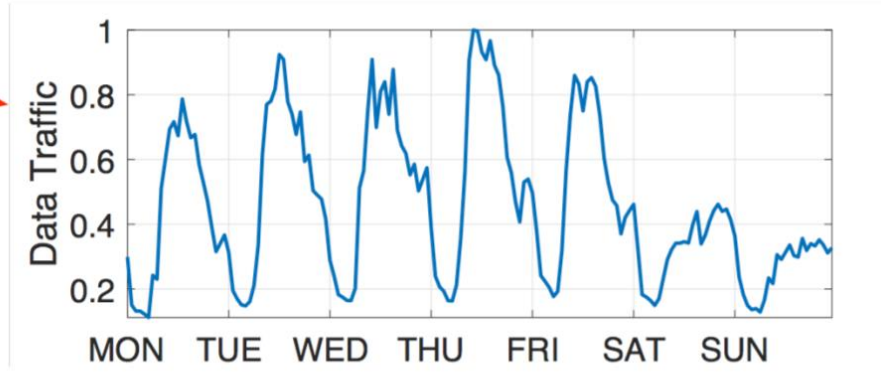
- **Demand Prediction**



(a) Base stations in Milan (11/01/2013 - 12/31/2013)



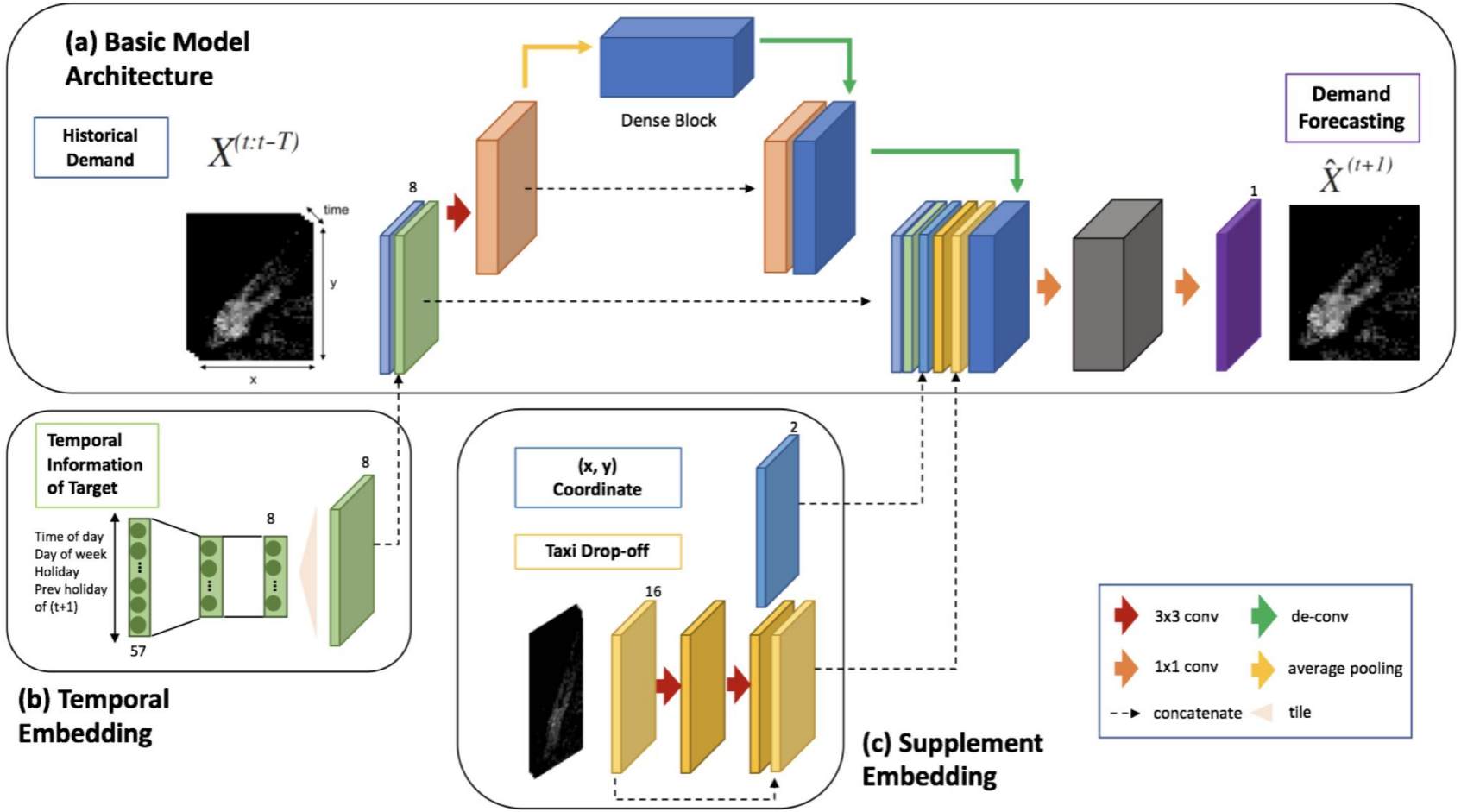
(b) One week of traffic in Centro Direzionale



(c) One week of traffic in Piazza Wagner

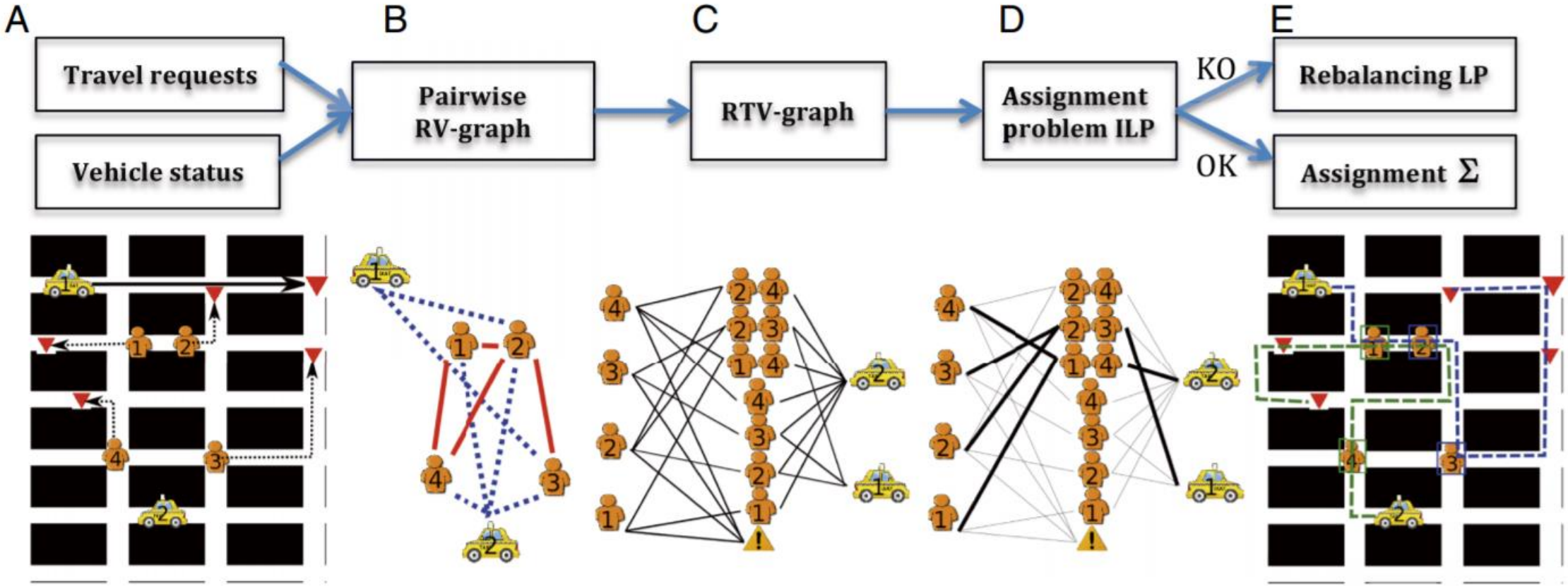
Research topic in AMoD system

- **Demand Prediction**
  : TG-Net (Kakao)



Research topic in AMoD system

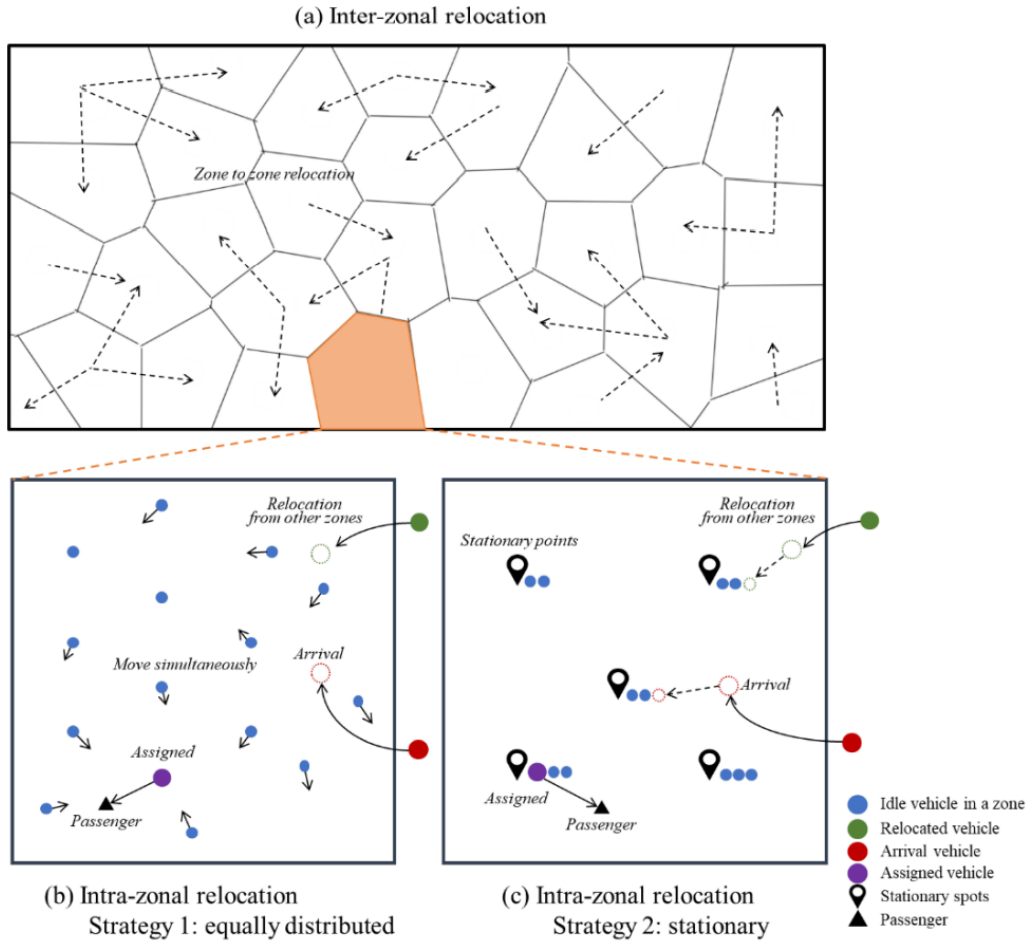
- **Dispatch**

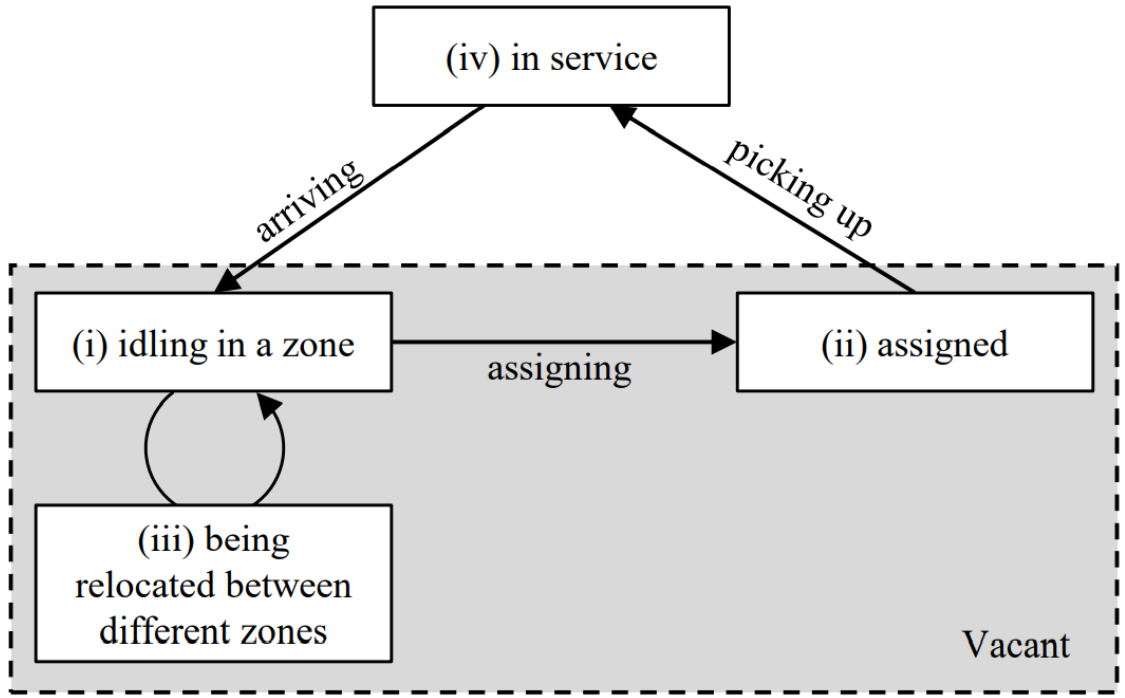


Alonso-Mora et al., 2016

Research topic in AMoD system

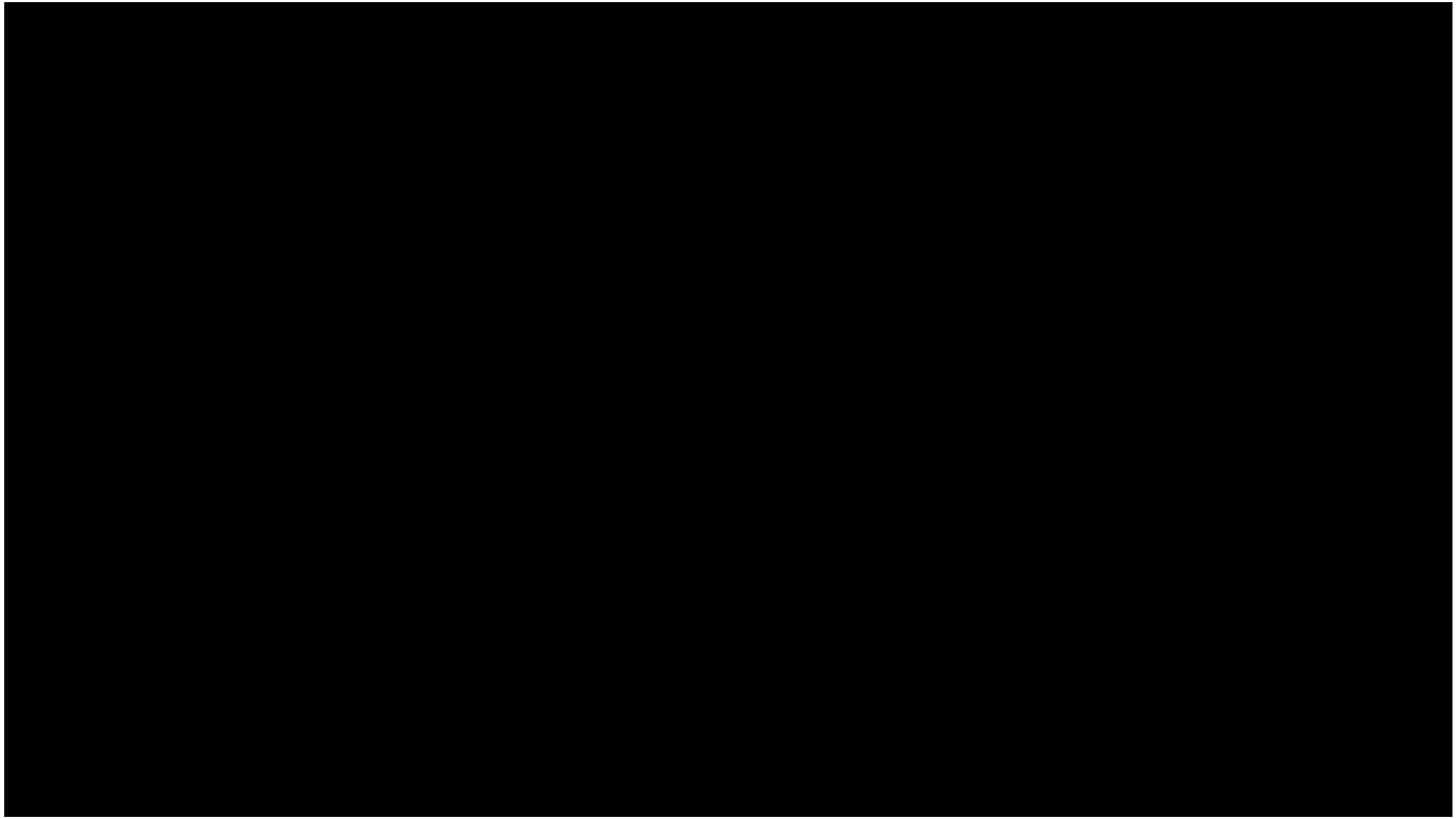
- **Relocation (Rebalancing, Repositioning)**





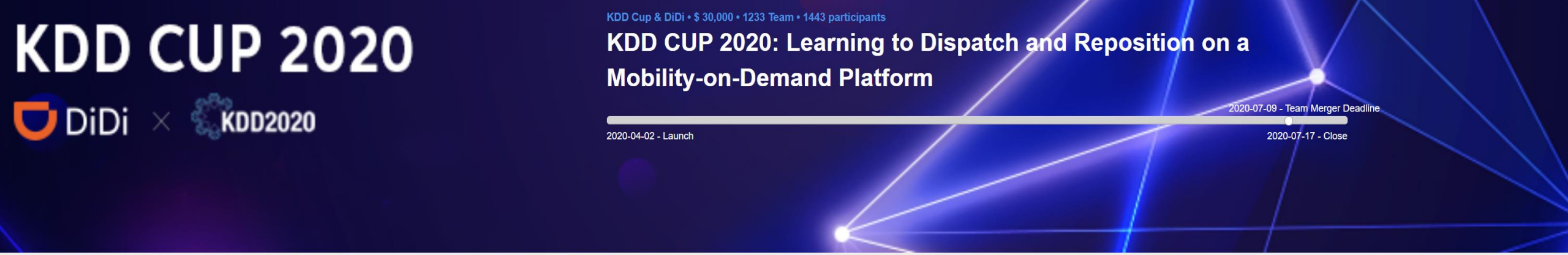
Construct Environment (Digital twin)

https://www.amodeus.science/



Construct Environment

https://github.com/wenjian0202/mod-abm-2.0



Introduction

Background

With the rising prevalence of smart mobile phones in our daily life, Mobility-on-Demand (MoD), or online ride-hailing platforms have emerged as a viable solution to provide more timely and personalized transportation service, led by such companies as DiDi, Uber, and Lyft. These platforms also allow idle vehicle vacancy to be more effectively utilized to meet the growing need of on-demand transportation, by connecting potential mobility requests to eligible drivers. A more efficient MoD system offers better user experience for both driver and passenger groups: Drivers would be able to generate higher income through reduced idle time. Passengers would enjoy shorter waiting time and higher fulfillment rate. The efficiency of an MoD system basically hinges on how well the supply and demand distributions are aligned in both spatial and temporal spaces. There are two important problems for optimizing the operational efficiency of an MoD platform through regulating the supply distribution to better align with the demand: vehicle repositioning and order dispatching (matching). Order dispatching matches idle drivers (vehicles) to open trip orders, transporting passengers (and the drivers) to the trip destinations. Vehicle repositioning is a proactive measure, by deploying idle vehicles to a specific location in anticipation of future demand at the destination or beyond.
This KDD Cup competition focuses on developing intelligent strategies to increase the drivers' income on an MoD platform through order dispatching and vehicle repositioning. Higher income for the drivers means better welfare, which in turn tends to translate better service for passengers.

Tasks

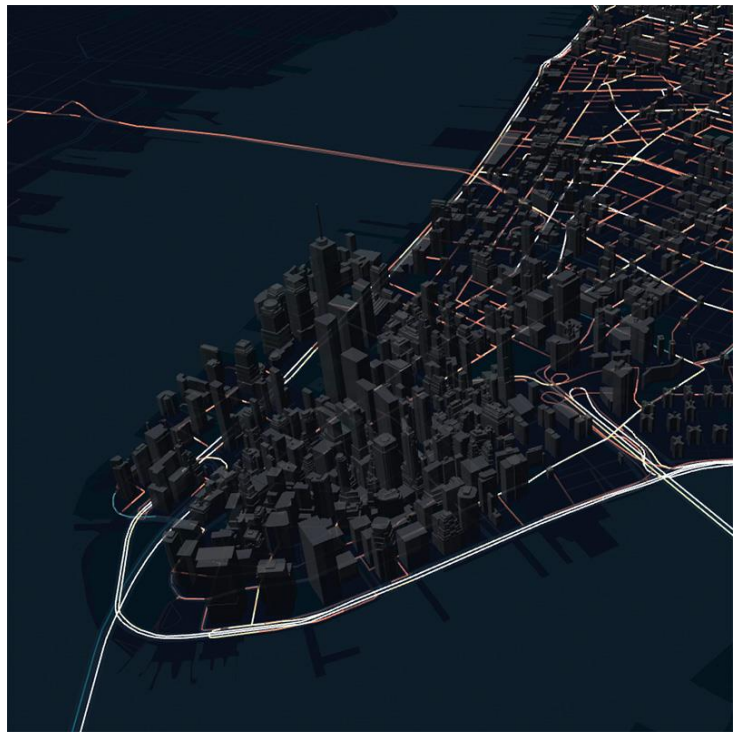
The challenge that the participants are to solve involves a combination of order dispatching (order matching) and vehicle repositioning (fleet management) on an MoD platform. The teams are to develop algorithms for either or both of these tasks. The algorithms are evaluated in a simulation environment that simulates the dynamics of an MoD platform. They will be asked to

Control Algorithms

- Dispatch
- Pricing
- Relocation
- Charging

Simulation

Real-world

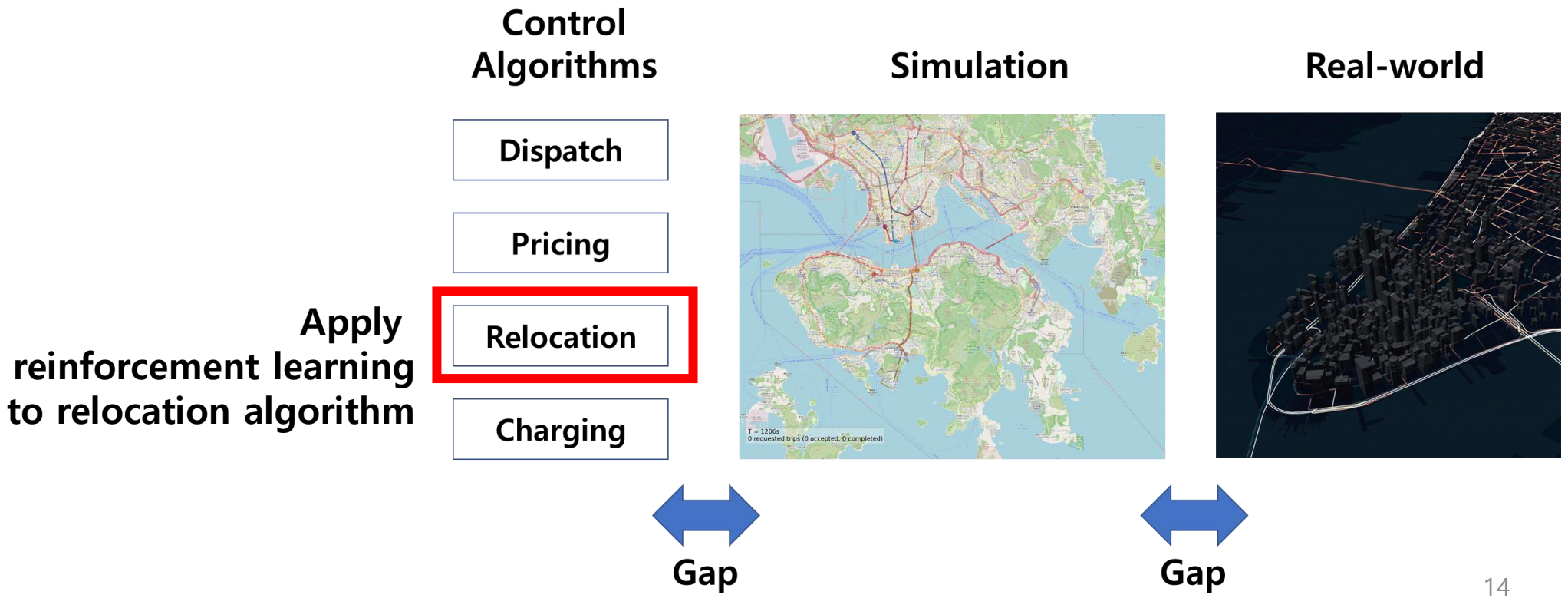


Gap

Gap

Paper Review

Graph Neural Network
Reinforcement Learning
for Autonomous Mobility-on-Demand Systems



Objective

- Propose **a deep reinforcement learning framework** to **control the rebalancing of AMoD systems** through graph neural networks
  - graph neural networks enable reinforcement learning agents to recover behavior policies that are significantly **more transferable, generalizable, and scalable**

Related work (Previous work)

1. **Rule-based heuristics**
   - Efficient
   - Rarely yield close-to-optimal solutions
2. **Model Predictive Control (MPC)**
   - Open-loop optimization problem is solved at each time step
   - Formulated into large-scale linear or integer programming problems
   - May not scale well for complex AMoD networks

Related work (Previous work)

3. Learning-based approaches

- RL-based decentralized approaches where **the action of each vehicle is determined independently** through a Q-learning policy
  → Efficient but lack of coordination
- **Cooperative multi-agent approach** for order dispatching and vehicle rebalancing using Deep Q-Networks and Proximal Policy Optimization
- Lacks a discussion on how to define neural network architectures able to **exploit the graph structure present in urban transportation networks**.

- Control-based vs Learning-based ??

Contribution

- **Compare control-based & learning-based models**
  - Use real-world trip data (Chengdu, China & New York, USA)

- **Transferable & Generalization**
  - Inter-city generalization (i.e., the agent is trained on one city and directly applied to another),
  - Service area expansion (i.e., the agent is trained on a specific sub-graph and directly applied to additional areas of the city)
  - Adaptation to irregular geographies (i.e., we measure the performance of the proposed framework when applied to arbitrarily complex transportation networks

Define Problem

Basic concepts of GNN

- **GNN**
  - Graphical representations of transportation systems are naturally defined by **node and edge properties** such as **demand in a region** or **travel time between regions**

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

$$\mathcal{V} = \{v_i\}_{i=1:N_v}$$

$$\mathcal{E} = \{e_k\}_{k=1:N_e}$$

invariant function taking as input (i) a D-dimensional feature description $\mathbf{x}_i$ for every node i (typically summarized in a $N_v \times D$ feature matrix $\mathbf{X}$), (ii) a representative description of the graph structure in matrix form $\mathbf{A}$ (typically in the form of an adjacency matrix), and produce an updated representation $\mathbf{x}'_i$ for all nodes in the graph.

  - GCN (Graph Convolutional Network)

$$\mathbf{X}' = f(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}\mathbf{X}\mathbf{W}\right)$$

Notations - 1

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.

Notations - 2

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}.

Notations - 3

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 $t+\tau_{ij}$.

Role of Operator

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.

Remark

- Travel times are given and independent of the control of the AMoD fleet
- The arrival process of passengers for each OD pair is a time-dependent Poisson process.

Three-Step Framework

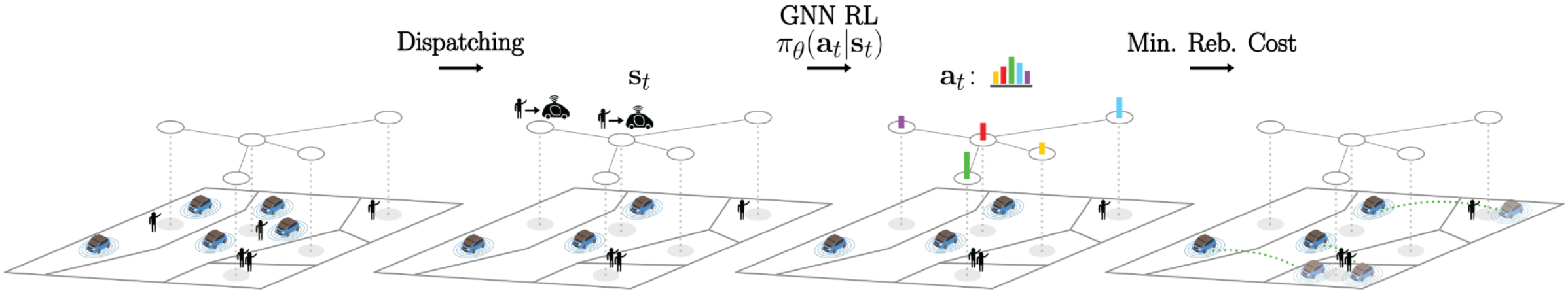


Fig. 2. An illustration of the three-step framework determining the proposed AMoD control strategy. Given the current distribution of idle vehicles (cars) and user transportation requests (stick figures), the control strategy is defined by: (1) dispatching idle vehicles to specific trip requests by solving a matching problem (thus, characterizing the current state $\mathbf{s}_t$ of the system), (2) computing an action $\mathbf{a}_t$ (i.e. the desired distribution of idle vehicles) using some policy π_θ, and (3) translate $\mathbf{a}_t$ into actionable rebalancing trips such that overall rebalancing cost is minimized.

Action: Computing the desired distribution of idle vehicles
→ Then translate action into actionable rebalancing trips

Three-Step Framework

A. A Three-Step Framework

As illustrated in Figure 2, we adopt a three-step decision-making framework similar to [10] 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 $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, (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.

Three-Step Framework
1 – Passenger matching (Dispatch)

$$\max_{\{x_{ij}^t\}_{i,j\in\mathcal{V}}} \quad \sum_{i,j\in\mathcal{V}} x_{ij}^t (p_{ij}^t - c_{ij}^t) \tag{2a}$$

$$\text{s.t.} \quad 0 \le x_{ij}^t \le d_{ij}^t, \ i,j \in \mathcal{V}, \tag{2b}$$

- x_{ij}^t - Passenger flow
- p_{ij}^t - Price (fee)
- c_{ij}^t - cost of traveling (function of travel time τ_{ij}^t)
- d_{ij}^t - Demand
- 최대한 많은 승객에게 돈을 많이 받으면서, 승객들의 travel time을 최소화 하도록 승객과 차량을 매칭

Three-Step Framework
2 – Determine desired idle vehicle distribution

The second step entails determining the desired idle vehicle distribution $\mathbf{a}^t_{\text{reb}} = \{\mathbf{a}^t_{\text{reb},i}\}_{i\in\mathcal{V}}$, where $\mathbf{a}^t_{\text{reb},i} \in [0,1]$ defines the percentage of currently idle vehicles to be rebalanced towards station i in time step t, and $\sum_{i\in\mathcal{V}} \mathbf{a}^t_{\text{reb},i} = 1$. With desired distribution $\mathbf{a}^t_{\text{reb}}$, denote $\hat{m}^t_i = \lfloor a^t_{reb,i} \sum_{i\in\mathcal{V}} m^t_i \rfloor$ as the number of desired vehicles, where m^t_i 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 ($\sum_{i\in\mathcal{V}} \hat{m}^t_i \leq \sum_{i\in\mathcal{V}} m^t_i$).

$\boldsymbol{a^t_{reb}}$

0.2	0.1	0.1
0.3	0.0	0.05
0.1	0.1	0.05

Three-Step Framework
3 – Execute Rebalancing

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

$$\min_{\{y_{ij}^t\}_{(i,j)\in\mathcal{E}}\in\mathbb{Z}_+^{|\mathcal{E}|}} \quad \sum_{(i,j)\in\mathcal{E}} c_{ij} y_{ij}^t \tag{3a}$$

$$\text{s.t.} \quad \sum_{j\neq i}(y_{ji}^t - y_{ij}^t) + m_i^t \geq \hat{m}_i^t, \ i \in \mathcal{V}, \tag{3b}$$

$$\sum_{j\neq i} y_{ij}^t \leq m_i^t, \ i \in \mathcal{V}, \tag{3c}$$

- y_{ij}^t - Rebalacing flows
- c_{ij}^t - cost of traveling (function of travel time τ_{ij}^t)

AMoD Rebalancing as Markov Decision Process : Action Space

$$\mathcal{M}_{\text{reb}} = (\mathcal{S}_{\text{reb}}, \mathcal{A}_{\text{reb}}, P_{\text{reb}}, r_{\text{reb}}, \gamma).$$

Action Space ($\mathcal{A}_{reb}$): Given a number of available vehicles $M_a \leq M$, and their current spatial distribution between the N_v stations, we consider the problem of determining the *desired idle vehicle distribution* $\mathbf{a}^t_{\text{reb}}$.

AMoD Rebalancing as Markov Decision Process : Reward

$$\mathcal{M}_{\text{reb}} = (\mathcal{S}_{\text{reb}}, \mathcal{A}_{\text{reb}}, P_{\text{reb}}, r_{\text{reb}}, \gamma).$$

Reward (r_{reb}): We define the reward function in the MDP from the perspective of an AMoD operator. That is, we express our objective so to recover behavior policies able to maximize travel demand satisfaction and provider profit, while minimizing the cost of unnecessary vehicle trips in the system. Specifically, each trip between two stations i, j will be characterized by a price p_{ij} and a cost c_{ij}, with $p_{ij} = 0$ in case of rebalancing trips. We can naturally express this objective through the following reward function:

$$r_{\text{reb}} = \sum_{i,j \in \mathcal{V}} x_{ij}^t (p_{ij}^t - c_{ij}^t) - \sum_{(i,j) \in \mathcal{E}} y_{ij}^t c_{ij}^t, \qquad (5)$$

AMoD Rebalancing as Markov Decision Process : State space

$$\mathcal{M}_{\text{reb}} = (\mathcal{S}_{\text{reb}}, \mathcal{A}_{\text{reb}}, P_{\text{reb}}, r_{\text{reb}}, \gamma).$$

State Space ($\mathcal{S}_{reb}$): We define the state in the rebalancing MDP to contain the information needed to determine proactive rebalancing strategies. Specifically, this will require knowledge of the structure of the transportation network through its adjacency matrix $\mathbf{A}$, together with additional station-level information by means of a feature matrix $\mathbf{X}$. In our experiments, we choose the adjacency matrix $\mathbf{A}$ to describe a transportation network where each station is connected to its spatially adjacent regions, such that all neighboring areas are connected by an edge.

AMoD Rebalancing as Markov Decision Process : State space

Moreover, we choose the feature matrix $\mathbf{X}$ to be a collection of three main sources of information. Firstly, we characterize the MoD system by the current availability of idle vehicles in each station $m_i^t \in [0, M], \forall i \in \mathcal{V}$. Given a planning horizon T, we also consider the *projected* availability of idle vehicles $\{m_i^{t'}\}_{t'=t,\dots,t+T}$, where this is estimated based on previously assigned passenger and rebalancing trips. Secondly, an effective rebalancing strategy will also depend on both current d_{ij}^t and estimated $\{\hat{d}_{ij}^{t'}\}_{t'=t,\dots,t+T}$ transportation demand between all stations. In this work, we assume to have access to a noisy and unbiased estimate of demand in the form of the rate of the underlying time-dependent Poisson process describing travel behavior in the system, although this could come from a predictive model such as [19]. Lastly, we also include provider-level information such as trip price p_{ij}^t and cost c_{ij}^t.

AMoD Rebalancing as Markov Decision Process : Dynamics (Transition Probability)

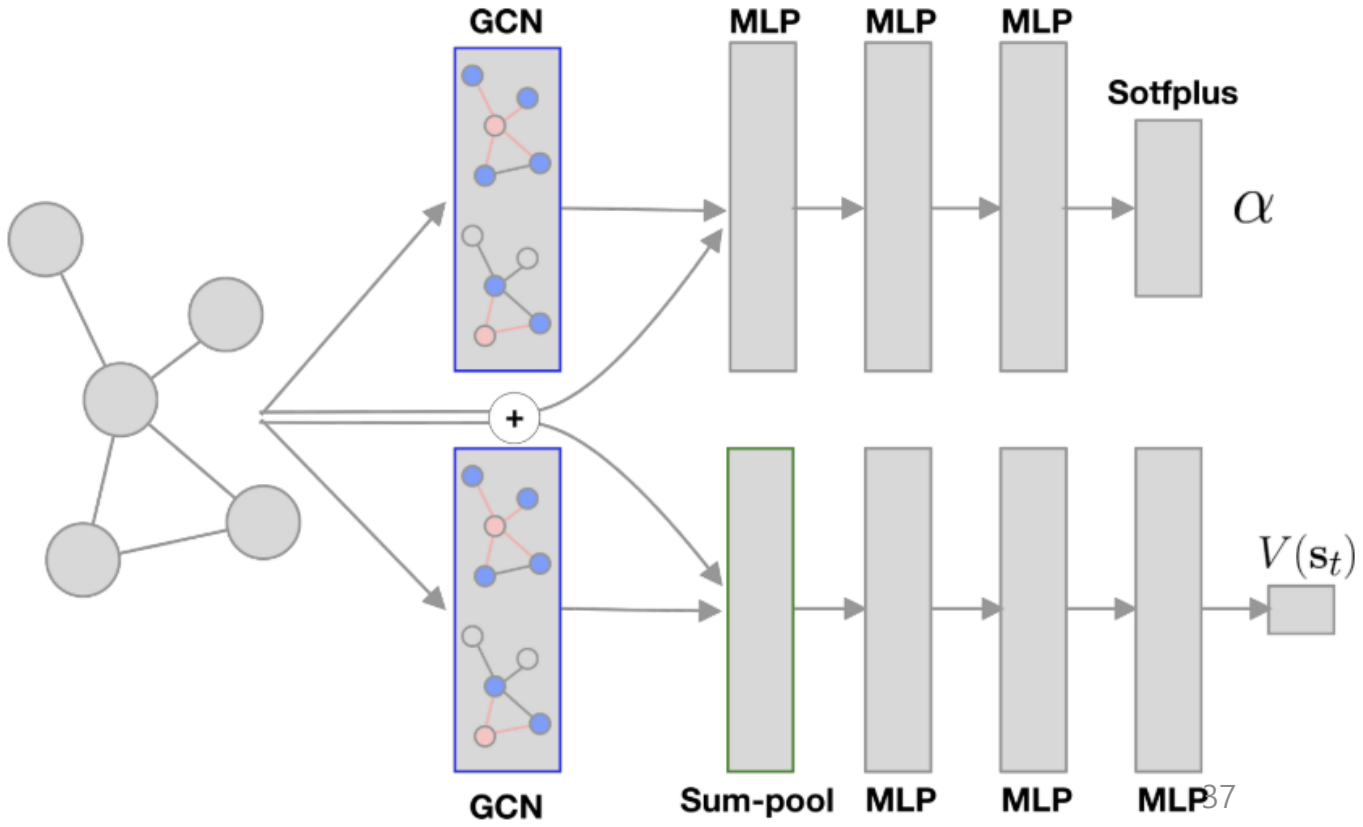
Dynamics (P_{reb}): The dynamics in the rebalancing MDP describe both the stochastic evolution of travel demand patterns, as well as how rebalancing decisions influence future state elements, such as the availability and distribution of idle vehicles. Specifically, the evolution of travel demand between stations d_{ij}^t is independent of the rebalancing action, and follows a time-dependent Poisson process (in our experiments, estimated from real trip travel data).

AMoD Rebalancing as Markov Decision Process : Dynamics (Transition Probability)

On the other hand, some of the state variable's transitions deterministically depend on the chosen action. For example, the estimated availability $\{m_i^{t'}\}_{t'=t,\dots,t+T}$ is uniquely defined as the sum of the current availability m_i^t together with the projected number of incoming vehicles at time t' (from both passenger and rebalancing flows), minus the vehicles currently chosen to be rebalanced. Finally, state variables related to provider information, such as trip price p_{ij}^t and cost c_{ij}^t are assumed to be externally decided and known beforehand (hence, independent from the actions selected by the behavior policy).

Graph Neural Network RL for Rebalancing : Actor-Critic algorithm

Having formally defined the AMoD rebalancing problem as an MDP, this section introduces the neural architecture for the policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ and the value function estimator $V_\phi(\mathbf{s}_t)$ characterizing the proposed Advantage Actor-Critic (A2C) algorithm [20]. Fig. 3 provides a schematic illustration.



Graph Neural Network RL for Rebalancing
: Policy – Dirichlet distribution

Policy. As introduced in Section III-B, a rebalancing action is defined as the desired distribution of idle vehicles across all N_v stations. Thus, in order for $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ to define a valid probability density over actions, we devise the output of our policy network to represent the concentration parameters $\alpha \in \mathbb{R}_+^{N_v}$ of a Dirichlet distribution, such that $\mathbf{a}_t \sim \text{Dir}(\mathbf{a}_t|\alpha) = \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ where $\text{Dir}(\cdot)$ denotes the Dirichlet distribution, and where the positivity of α is ensured by e.g., a Softplus nonlinearity.

Graph Neural Network RL for Rebalancing : Policy – Dirichlet distribution

분포 [편집]

2 이상의 자연수 k와 양의 상수 $\alpha_1, \cdots, \alpha_k$에 대하여, 디리클레 분포의 확률 밀도 함수는 다음과 같이 정의된다. 실수값 $x_1, \cdots, x_k$가 모두 양의 실수이며 $\sum_{i=1}^{k} x_i = 1$을 만족할 때

$$f(x_1, \cdots, x_k; \alpha_1, \cdots, \alpha_k) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$$

의 값을 가지며, 그 외의 경우는 0의 값을 가진다. 이때 $\alpha = (\alpha_1, \cdots, \alpha_k)$이며, $\mathrm{B}(\alpha)$는 정규화 상수로서 다음의 값을 가진다.

$$\mathrm{B}(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)} \quad (\Gamma\text{는 감마 함수})$$

디리클레 분포에서 $k = 2$인 경우 베타 분포가 된다.

https://ko.wikipedia.org/wiki/%EB%94%94%EB%A6%AC%ED%81%B4%EB%A0%88_%EB%B6%84%ED%8F%AC

Experiments

Experiment setting

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 km×4 km) and Chengdu (7 a.m. – 10 a.m., with a size of 10 km×10 km), respectively. The studied areas in both case studies are divided into grid-like blocks, each of which represents a station.

Performance

- Best performance는 MPC,
- real-time applications for large scale networks 불가능

TABLE I

SYSTEM PERFORMANCE ON CHENGDU 4 × 4 NETWORK

	Reward (%Dev. MPC-tri-level)	Served Demand	Rebalancing 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 II

SYSTEM PERFORMANCE ON NEW YORK 4 × 4 NETWORK

	Reward (%Dev. MPC-tri-level)	Served Demand	Rebalancing 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

Inter-city Portability

To assess the transferability and generalization capabilities of A2C-GNN, we also study the extent to which policies can be trained on one city and later applied to the other *without further training* (i.e., zero-shot). Specifically, in this simulation experiment, we select the pre-trained policy on Chengdu data, and examine its zero-shot performance when deployed in New York's transportation network, and vice-versa.

TABLE I

SYSTEM PERFORMANCE ON CHENGDU 4 × 4 NETWORK

	Reward (%Dev. MPC-tri-level)	Served Demand	Rebalancing 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 II

SYSTEM PERFORMANCE ON NEW YORK 4 × 4 NETWORK

	Reward (%Dev. MPC-tri-level)	Served Demand	Rebalancing 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

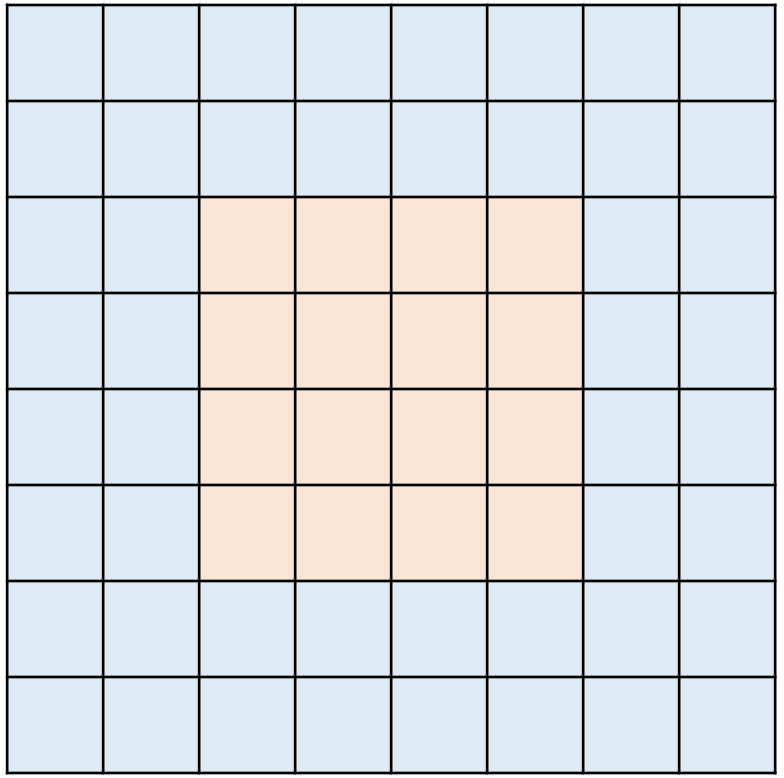
Service Area Expansion

TABLE III

SYSTEM PERFORMANCE ON NEW YORK 8 × 8 NETWORK

	Reward (%Dev. MPC-standard)	Served Demand	Rebalancing Cost ($)
ED	41,930 (-24.4%)	13,028	11,023
A2C-GNN-0Shot	46,516 (**-15.1%**)	13,974	10,083
A2C-GNN	47,843 (**-12.6%**)	14,165	12,165
MPC-standard	54,737 (0%)	16,275	10,389

Table III show that A2C-GNN-0Shot is only 2.5% less profitable and satisfies 1.3% fewer customers when compared to its fully-retrained counterpart (A2C-GNN)



Irregular Geographies

TABLE IV

SYSTEM PERFORMANCE ON THE IRREGULAR 16-DIMENSIONAL TOPOLOGY (NEW YORK)

	Reward (%Dev. MPC-tri-level)	Served Demand	Rebalancing Cost ($)
ED	7,900 (-27.9%)	2,431	3,451
A2C-GNN	9,981 (**-8.9%**)	2,531	1,371
MPC-tri-level	10,955 (0%)	2,527	382
MPC-standard	10,127 (-7.6%)	2,376	379

We now investigate how well the pre-trained A2C-GNN can be applied to arbitrary, non-grid-like transportation networks. Specifically, we select 16 stations defining a *disjoint* service area, thus not representable as a contiguous grid

Network Granularity & Computation Efficiency

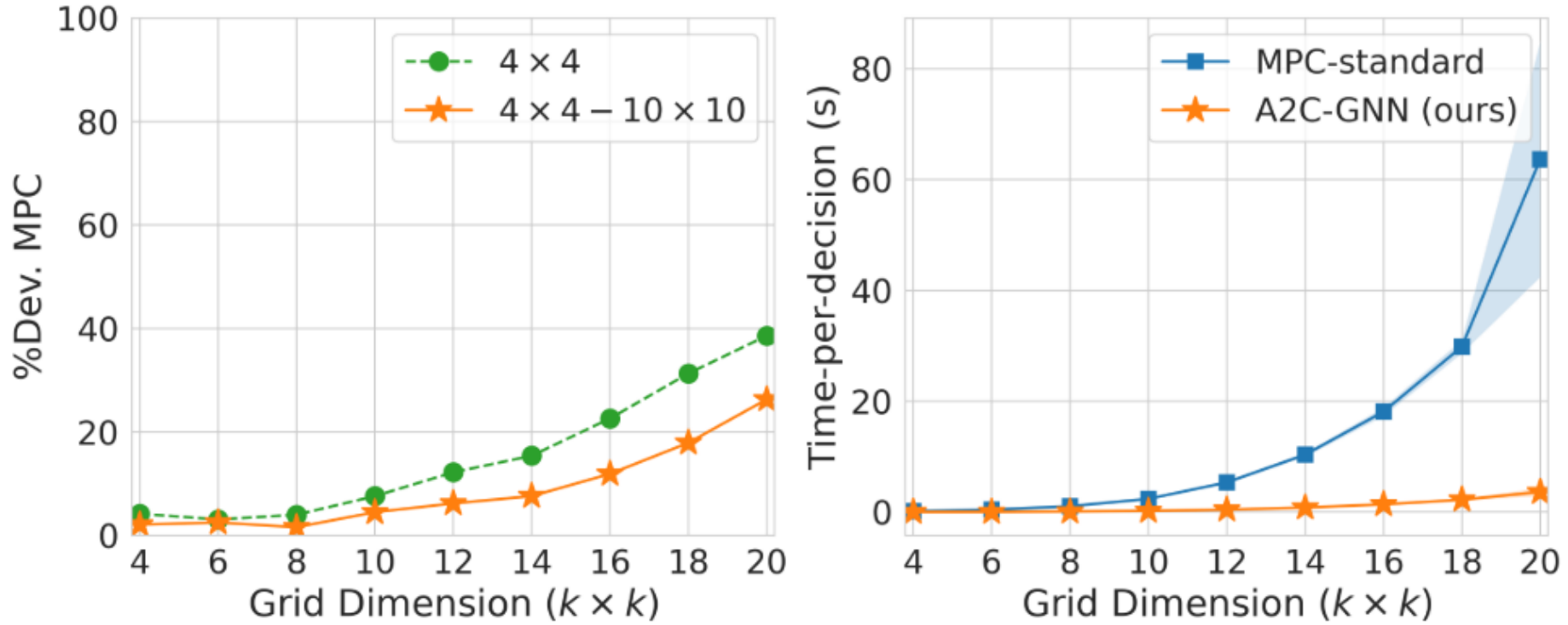


Fig. 4. Left: System performance (Percentage Deviation from MPC-standard) for agents trained either on a single granularity (4×4) or across granularities (4×4 - 10×10), Right: Comparison of computation times between A2C-GNN and MPC-standard.

감사합니다

rnt5306@gmail.com