Benchmarking Reinforcement Learning for Network-level Coordination of Autonomous Mobility-on-Demand Systems Across Scales

Luigi Tresca*†, Daniele Gammelli†, James Harrison‡, Gioele Zardini†§, and Marco Pavone†
*Energy Department, Politecnico di Torino, Turin, Italy, luigi.tresca@polito.it
†Department of Aeronautics and Astronautics, Stanford University, USA, {ltresca, gammelli, zardini, pavone}@stanford.edu
‡Google DeepMind, San Francisco, CA, USA, jamesharrison@google.com
§Laboratory for Information and Decision Systems, MIT, Cambridge, MA, USA, gzardini@mit.edu

***Abstract*—As urbanization intensifies, relying on private cars for private mobility becomes increasingly unsustainable, prompting the exploration of alternative transit solutions. This paper focuses on autonomous mobility-on-demand (AMoD) systems, which leverage fleets of autonomous vehicles (AVs) controlled via sophisticated algorithms to deliver on-demand mobility services. Such systems are promising given their ability to react and predict fluctuating spatio-temporal travel demand patterns, but raise several computational challenges associated with large fleets and the complex networks on which they operate. This paper introduces a novel decision-making framework that integrates optimization strategies with data-driven techniques. In particular, our approach enables effective learning of rebalancing policies via deep reinforcement learning, ensuring close-to-optimal performance, computational tractability, and generizability across different urban settings. Furthermore, it provides an interface for practitioners to learn and evaluate policies with custom fidelity requirements, all the way from heuristic-based approaches, to microscopic traffic simulators. Finally, we showcase the properties of the framework on the real-world case studies of NYC, USA, Chengdu, China, and Luxembourg.**

I. Introduction

Recent urbanization trends have resulted in increased travel and related externalities, with cities accounting for over 78% of the world's energy consumption, and for over 60% of the global greenhouse gas emissions (30% of which is produced by transportation, in the USA) [1]. Congestion levels worldwide are escalating, causing 8.8 billion hours of extra travel time in the USA alone (equivalent to over a week of vacation for the average US commuter). In this context, relying on private cars for personal mobility is becoming increasingly impractical and unsustainable. Ride-hailing services have emerged as an alternative, offering mobility-on-demand (MoD) services, raising concerns related to the exploitation of public resources, equity, profitability, and, importantly, scalability. In particular, the travel demand for such services is spatio-temporally asymmetrically distributed (e.g., commuting toward downtown in the morning), making the overall operations *imbalanced* and extremely sensitive to disturbances [2]. In this context, technological advances in the field of autonomous driving offer a new mobility paradigm: autonomous mobility-on-demand (AMoD). An AMoD system consists of a fleet of autonomous vehicles (AVs) that pick up passengers, and transport them to their destination. A manager controls the fleet by simultaneously assigning passengers to AVs, routing them, and rebalancing the fleet by relocating customer-free AVs to realign their geographical distribution with transportation demand (thereby solving the aforementioned issue). AMoD services promise two main benefits: they increase the supply of drivers to match increasing demand, and drastically reduce transportation costs [2]. However, such services also entail controlling thousands of AVs in complex, congested networks, raising numerous challenges at the interface of computational scalability of control schemes and system performance. In this work, we propose a hierarchical decision-making framework to centrally control AMoD systems. Specifically, our framework blends optimization-based methods and data-driven approaches by exploiting the main strengths of graph neural networks (GNNs), reinforcement learning (RL), and classical operations research tools. We show that our framework exhibits a number of desirable properties, including computational tractability, generalizability, and close-to-optimal performance. Furthermore, we show the ability of the proposed approach to interface with policy evaluation tools of varying fidelity, starting from heuristic-based ones, all the way to microscopic traffic simulators.

Related Work. Existing literature on the real-time coordination of AMoD systems can be classified into three categories. The first category applies simple rule-based heuristics [3, 4] that, although efficient, can rarely yield close-to-optimal solutions. The second category designs Model Predictive Control (MPC) approaches based on network flow models [5, 6], whereby an embedded open-loop optimization problem is solved at each time step to yield a sequence of control actions over a receding horizon, but only the first control action is executed. These embedded optimization problems are typically formulated into large-scale linear or integer programming problems, which may not scale well for complex AMoD networks. The third and most relevant category for this work employs learning-based approaches to devise efficient algorithms without significantly compromising optimality [7, 8, 9]. Guériau et al. [7] developed RL-based decentralized approaches through a Q-learning policy; Holler et al. [8] developed a cooperative multi-agent approach for order dispatching and vehicle rebalancing using Deep Q-Networks and Proximal Policy Optimization; Fluri et al. [9] developed a cascaded Q-learning approach to operate AMoD systems in a centralized manner. Overall, although the afore-

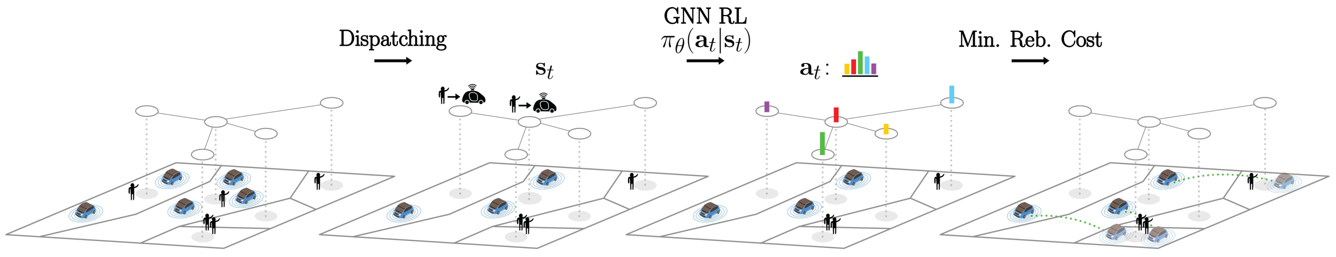

Fig. 1. An illustration of the three-step framework for the proposed AMoD control strategy. Given the current distribution of idle vehicles and travel requests (stick figures), the control strategy is defined by: (1) dispatching idle vehicles to specific trip requests by solving a matching problem, (2) computing an action (i.e., the desired distribution of idle vehicles) using some policy, and (3) translate it into actionable rebalancing trips to minimize the overall rebalancing cost.

mentioned RL-based works cover a wide range of algorithms, there lacks a discussion on how to (i) combine the benefits of learning-based and optimization-based methods, and (ii) define neural network architectures able to exploit the graph structure present in urban transportation networks.

Contributions. Concretely, the contributions of this work are threefold, and build on previous efforts presented in [10, 11, 12]. First, we present a novel hierarchical policy framework that leverages the specific strengths of direct optimization and graph network-based RL. Second, we show that our approach is highly performant, scalable, and robust to changes in operating conditions and network topologies, across simulators of different fidelity. Crucially, we demonstrate that our approach outperforms classical optimization-based techniques, domain-specific heuristics, and pure end-to-end RL. Finally, motivated by the need to democratize research efforts in this area, we move the first steps toward the creation of publicly available benchmarks, datasets, and simulators for network-level coordination of MoD systems and release code[1] to (i) provide openly accessible simulation platforms, and (ii) create a common validation process and allow direct comparison between different methodologies.

II. Methodology

In this section, we present a methodology to control the operations of AMoD systems. In particular, we first introduce the notation and provide a problem description, and then describe a three-step approach to solve it integrating optimization and data-driven techniques.

Problem Definition. An AMoD operator coordinates M robo-taxis to provide on-demand mobility services on a transportation network represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of stations (e.g., pick-up or drop-off locations) and $\mathcal{E}$ represents the set of links connecting two adjacent stations. Let $N_v = |\mathcal{V}|$ denote the number of stations. The time horizon is discretized into a set of discrete intervals $\mathcal{I} = \{1, 2, \dots, T\}$ of a given length ΔT. AVs traveling between station $i \in \mathcal{V}$ and station $j \neq i \in \mathcal{V}$ starting from each time step t are controlled by the operator to follow the shortest path, with a travel time of $\tau_{ij}^t \in \mathbb{Z}_+$ time steps and a travel cost of c_{ij}^t (e.g., as a function of travel time). Passengers make trip requests at each time step. The requests with origin-destination (OD) pair $(i,j) \in \mathcal{V} \times \mathcal{V}$ submitted at time step $t \in \mathcal{I}$ are characterized by demand d_{ij}^t and price p_{ij}^t. With such trip requests, the operator dynamically matches passengers to vehicles, and the matched vehicles will deliver passengers to their destinations. Idle vehicles not matched with any passengers will be controlled by the operator to either stay at the same station or rebalance to other stations. We denote $x_{ij}^t \in \mathbb{N}$ 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, and $y_{ij}^t \in \mathbb{N}$ as the rebalancing flow, i.e., the number of vehicles rebalancing from station i to station j at time step t. Thus, the goal of a service operator is to compute passenger and rebalancing flows to optimize a given performance metric.

Proposed Framework. We formulate the AMoD control problem as a hierarchical, three-step decision-making process: demand-vehicle matching, vehicle rebalancing via RL, and post-processing (Figure 1). This three-step framework, as shown in the rest of this section, has the advantage of reducing the action space from N_v^2 to N_v, since the learned policy defines an action at each node as opposed to along each OD pair (as in most of the literature). In the following, we provide a detailed description of the framework. First, when matching, the operator assigns vehicles to customers and obtains passenger flows $\{x_{ij}^t\}_{i,j \in \mathcal{V}}$ by solving the following assignment problem:

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

$$\text{s.t.} \quad 0 \le x_{ij}^t \le d_{ij}^t, \; i,j \in \mathcal{V}, \tag{1b}$$

$$\sum_{j \in \mathcal{V}} x_{ij}^t \le M_i^t, \; i \in \mathcal{V}, \tag{1c}$$

where the objective function (1a) represents the profit of passenger assignment calculated as the difference between revenue and cost, the constraint (1b) ensures that the passenger flow is non-negative and upper-bounded by the demand, and the constraint (1c) represents that the total passenger flow does not exceed the number of vacant vehicles M_i^t at station i at time step t. Note that the constraint matrix of the assignment problem (1) is totally unimodular [13], hence the resulting passenger flows are integral as long as the demand is integral.

Second, the RL[2] step aims to determine the desired idle

[1]Code available at: https://github.com/StanfordASL/RL4AMOD

[2]Refer to the Appendix A for a detailed overview of common techniques.

vehicle distribution $\mathbf{a}_{\text{reb}}^t = \{a_{\text{reb},i}^t\}_{i \in \mathcal{V}}$, where $a_{\text{reb},i}^t \in [0,1]$ defines the percentage of currently idle vehicles to be rebalanced towards station i in time step t, and $\sum_{i \in \mathcal{V}} a_{\text{reb},i}^t = 1$. With desired distribution $\mathbf{a}_{\text{reb}}^t$, denote $\hat{m}_i^t = \lfloor a_{\text{reb},i}^t \sum_{i \in \mathcal{V}} m_i^t \rfloor$ as the number of desired vehicles, where m_i^t represents the actual number of idle vehicles after matching 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 integral and always available ($\sum_{i \in \mathcal{V}} \hat{m}_i^t \leq \sum_{i \in \mathcal{V}} m_i^t$).

Third, the post-processing step converts the desired distribution into actionable rebalancing flows $\{y_{ij}^t\}_{i \neq j \in \mathcal{V}}$ by solving a minimal rebalancing-cost problem:

$$\min_{\{y_{ij}^t\}_{i \neq j \in \mathcal{V}} \in \mathbb{N}^{|\mathcal{V}| \times (|\mathcal{V}|-1)}} \quad \sum_{i \neq j \in \mathcal{V}} c_{ij}^t y_{ij}^t \tag{2a}$$

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

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

where the objective function (2a) represents the rebalancing cost, constraint (2b) ensures that the resulting number of vehicles is close to the desired one, and constraint (2c) ensures that the total rebalancing flow from a region is upper-bounded by the number of idle vehicles in that region.

III. Results

In this section, we showcase the properties of the described methodology by studying real-world case studies. In particular, our exploration is fourfold. First, we evaluate the performance of our three-step controller with different metrics and compare it with other baselines, including heuristics and optimization-based policies. Second, we evaluate the ability of the learned policies to generalize to new environments. Third, we provide insights about the advantages in terms of computational effort when comparing our policy with optimization-based strategies. Finally, we showcase the ability of the proposed framework to interface with simulators with varying fidelity levels, by exploring the evaluation of policies leveraging microscopic simulators. Please refer to Appendix C for a detailed description of the experimental setup.

A. Performance properties and baseline comparisons

To explore the performance of the proposed scheme, we base our experiments on the morning commute demand in popular areas of the city of New York, USA, and the city of Chengdu, China. Both cities are divided into a 16-dimensional (i.e. 4×4) grid, where each grid block represents a station to which people can commute. The case study is generated leveraging historic demand datasets, which are converted to obtain the demand, travel time, and prices for origin-destination (OD) pairs [14, 15]. The performance of the learned policy is evaluated by three Key Performance Indicators (KPIs) in addition to the reward function, including (i) the *served demand*, defined as the total number of trips satisfied by the AMoD control strategy, (ii) the *rebalancing cost*, defined as the overall cost induced on the system by the rebalancing policy, and (iii) the *percentage deviation* from MPC performance having oracle information of future system states (%Dev. MPC). The performances of the policies for NYC and Chengdu are reported in Tables I and II, respectively. The proposed RL-based methodology can learn rebalancing policies able to achieve close-to-optimal system performance on both tasks. Specifically, RL performance is only 4.3% (New York) and 4.3% (Chengdu) away from the oracle MPC performance. Moreover, the RL-based policy is able to achieve more than 9% (New York) and 17% (Chengdu) in profit (reward) improvement while reducing the rebalancing cost by about 37% (New York) and 70% (Chengdu) when compared to the heuristic policy (ED).

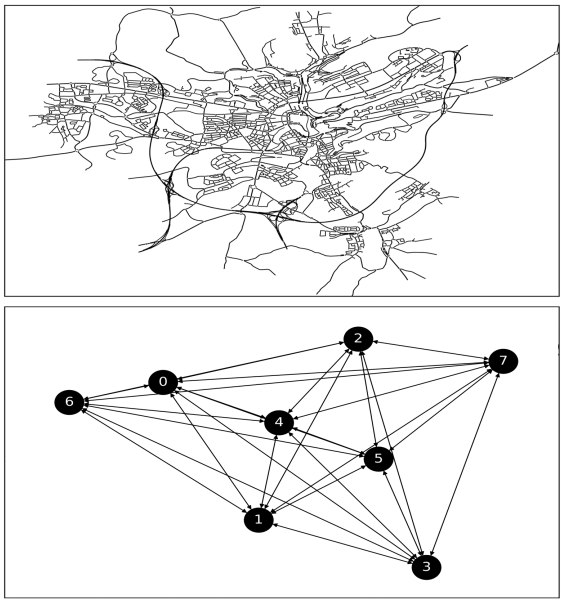

Fig. 2. Luxembourg network, including the original network (top) and the aggregated network described as a complete digraph (bottom).

TABLE I
System Performance on New York Macroscopic Simulation

	Reward (%Dev. MPC-oracle)	Served Demand	Rebalancing Cost ($)
ED	30,746 (-13.4%)	8,770	7,990
RL (ours)	33,886 (**-4.3%**)	8,772	5,038
MPC-oracle	35,356 (0%)	8,968	4,296
RL-0Shot	33,397 (**-5,7%**)	8,628	4,743

B. Generalizability to new environments

To assess the generalization capabilities of the RL policy, we measure transfer performance in the case of (i) intercity portability, (ii) service area expansion, (iii) irregular

TABLE II
System Performance on Chengdu Macroscopic Simulation

	Reward (%Dev. MPC-oracle)	Served Demand	Rebalancing Cost ($)
ED	12,538 (-26,8%)	41,189	3,397
RL (ours)	15,167 (**-9,8%**)	40,578	1,063
MPC-oracle	16,702 (0.0%)	44,662	1,162
RL-0Shot	14,791 (**-12,3%**)	40,646	1,467

geographies, and (iv) changes in network granularity. Given the space limitations, we discuss (i) in the main text and refer the reader to Appendix C-C for (ii)-(iv). Specifically, we study the extent to which policies can be trained in one city and later applied to the other *without further training* (i.e., zero-shot). Without any fine-tuning, the only way for an effective rebalancing policy to emerge is if the agent has learned a high-level, abstract, and generalizable understanding of the system dynamics. Tables II and I show the adaptation performance (RL-0Shot) when the NYC policy is applied in Chengdu, and vice-versa, highlighting an interesting degree of inter-city portability of both the rebalancing policies. In particular, despite the significantly different mobility patterns and topologies, the zero-shot policy shows only a slight drop in performance when compared to its fully re-trained counterpart: 1.4% for New York and 2.5% for Chengdu, respectively.

C. Computational analysis

We compare the time required to compute a single rebalancing decision for both the RL and the MPC approaches at different transportation network dimensions, as reported in Figure 5. The results show how the computational complexity of RL-based policies scales linearly in the number of nodes and graph connectivity, as opposed to optimization-based ones which scale super-linearly in the number of edges [16].

D. Higher fidelity for policy evaluations via micro-simulations

Lastly, we apply the proposed RL-based framework to a high-fidelity scenario developed within the SUMO simulator [17]: a microscopic and continuous traffic simulation package. In particular, we use a mesoscopic approach [18] to model the traffic flow in the transportation network. We model the AMoD system as a taxi fleet operating in the city of Luxembourg, incorporating both exogenous traffic and passenger demand. This approach aims to close the gap between simulation and real-world traffic scenarios for the AMoD system. Moreover, enhancing the fidelity of the traffic environment also increases the stochasticity of the simulation. In this context, the ability of learning-based methods to handle stochastic environments makes them particularly suitable for addressing the related decision-making challenges. We evaluate the performance of the learning-based approach using the same KPIs of the previous scenario, as reported in Table III. In addition to the baselines used in the previous scenario, we consider another heuristic policy (P1). Crucially, the micro-simulation results confirm the takeaways from the previous sections, with the RL-based policy achieving close-to-optimal system profit, i.e., only 3.9% away from Oracle performance. As in the previous scenario, the RL policy can reduce the rebalancing cost by about 44% when compared to the ED policy, highlighting a more efficient strategy, while outperforming the P1 policy in terms of satisfied travel demand. Notably, both the ED and RL policies fully satisfy the demand while the MPC does not serve some unprofitable trips.

Figure 3 further compares the RL rebalancing strategy (blue bar) with the ED policy (yellow bar) in satisfying the demand (red) per each region of the network. The incoming vehicles in a region are split into rebalanced vehicles (striped bar) and vehicles that have served a trip demand. Figure 3 shows how the RL-based policy is able to better close the gap between the travel demand and the number of vehicles that are available in a region by reducing the number of unnecessary rebalancing trips, thus highlighting a more forward-thinking approach to vehicle distribution. On the other hand, in accordance with the findings presented in Table III, the ED policy is myopic to future incoming vehicles in the region and proposes a rebalancing strategy that is characterized by more frequent rebalancing and higher associated costs.

TABLE III
SYSTEM PERFORMANCE ON LUXEMBOURG MESOSCOPIC SIMULATION

	Reward (%Dev. MPC-oracle)	Served Demand	Rebalancing Cost ($)
ED	20,17 (-20,1%)	100%	8,85
P1	29,48 (-22,7%)	79%	3,38
RL (ours)	24,25 (**-3,9%**)	100%	5,00
MPC-oracle	25,24 (0%)	95%	2,40

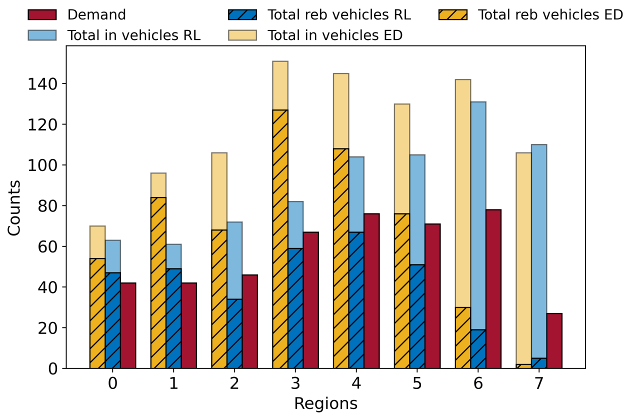

Fig. 3. Bar plot showing the trip demand (dark red) and number of incoming vehicles per region in the Luxembourg case study. The proportion of rebalanced vehicles (striped bar) is superposed to the total number of incoming vehicles. The proposed RL policy (blue bar) is compared to the ED policy (yellow bar).

IV. OUTLOOK

This work paves the way for four main future investigations. First, we would like to continue assessing the potential of the proposed framework with increasing levels of fidelity, all the way to employing microscopic traffic simulators. Second, we would like to extend the analysis of generalizability across fidelity levels (e.g., macroscopic to mesoscopic and microscopic), and characterize the arising trade-offs in performance and computation. Third, we would like to involve multiple communities by releasing benchmarks, in a quest to democratize the algorithmic developments for the presented problems. Finally, we aim to expand the portfolio of applications that could benefit from the presented approach, including network-based problems at large.

ACKNOWLEDGMENTS

D. Gammelli and G. Zardini are supported by the NASA University Leadership Initiative (grant #80NSSC20M0163). This article solely reflects the opinions and conclusions of its authors and not any NASA entity.

References

[1] D. o. E. a. S. A. United Nations, "68% of the world population projected to live in urban areas by 2050, says un," UN, Tech. Rep., 2021. [Online]. Available: https://www.un.org/development

[2] G. Zardini, N. Lanzetti, M. Pavone, and E. Frazzoli, "Analysis and control of autonomous mobility-on-demand systems: A review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 633–658, 2022. [Online]. Available: https://www.annualreviews.org/doi/abs/10.1146/annurev-control-042920-012811

[3] M. Hyland and H.-S. Mahmassani, "Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 278–297, 2018.

[4] M. W. Levin, K. M. Kockelman, S. D. Boyles, and T. Li, "A general framework for modeling shared autonomous vehicles with dynamic network-loading and dynamic ride-sharing application," *Computers, Environment and Urban Systems*, vol. 64, pp. 373 – 383, 2017.

[5] R. Zhang and M. Pavone, "Control of robotic Mobility-on-Demand systems: A queueing-theoretical perspective," *Int. Journal of Robotics Research*, vol. 35, no. 1–3, pp. 186–203, 2016.

[6] R. Iglesias, F. Rossi, K. Wang, D. Hallac, J. Leskovec, and M. Pavone, "Data-driven model predictive control of autonomous mobility-on-demand systems," in *Proc. IEEE Conf. on Robotics and Automation*, Brisbane, Australia, May 2018.

[7] M. Guériau, F. Cugurullo, R. Acheampong, and I. Dusparic, "Shared Autonomous Mobility on Demand: A learning-based approach and its performance in the presence of traffic congestion," vol. 12, no. 4, pp. 208–218, 2020.

[8] J. Holler, R. Vuorio, Z. Qin, X. Tang, Y. Jiao, T. Jin, S. Singh, C. Wang, and J. Ye, "Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem," in *IEEE Int. Conf. on Data Mining*, 2019.

[9] 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.

[10] D. Gammelli, K. Yang, J. Harrison, F. Rodrigues, F. C. Pereira, and M. Pavone, "Graph neural network reinforcement learning for autonomous mobility-on-demand systems," in *Proc. IEEE Conf. on Decision and Control*, 2021. [Online]. Available: https://arxiv.org/abs/2104.11434

[11] D. Gammelli, K. Yang, J. Harrison, F. Rodrigues, F. Pereira, and M. Pavone, "Graph meta-reinforcement learning for transferable autonomous mobility-on-demand," in *ACM Int. Conf. on Knowledge Discovery and Data Mining*, 2022. [Online]. Available: https://arxiv.org/abs/2202.07147

[12] D. Gammelli, J. Harrison, K. Yang, M. Pavone, F. Rodrigues, and P. C. Francisco, "Graph reinforcement learning for network control via bi-level optimization," in *Int. Conf. on Machine Learning*, 2023.

[13] G. L. Nemhauser, *Integer programming and combinatorial optimization*. Springer, vol. 191.

[14] Taxi & Limousine Commission. (2013) New York City Taxi & Limousine Commission Trip Record Data. See https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[15] Didi Chuxing Technology Co. (2016) Didi Chuxing Gaia Initiative. Available at https://outreach.didichuxing.com/research/opendata/en/.

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

[17] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[18] N. G. Eissfeldt, "Vehicle-based modelling of traffic . theory and application to environmental impact modelling," Ph.D. dissertation, Universität zu Köln, 2004.

[19] D. Gammelli, I. Peled, F. Rodrigues, D. Pacino, and F. Pereira, "Estimating latent demand of shared mobility through censored gaussian processes," *Transportation Research Part C: Emerging Technologies*, vol. 120, 2020.

[20] T.-N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Int. Conf. on Learning Representations*, 2017.

[21] J. Rawlings and D. Mayne, *Model predictive control: Theory and design*. Nob Hill Publishing, 2013.

[22] T. Van de Wiele, D. Warde-Farley, A. Mnih, and V. Mnih, "Q-learning in enormous action spaces via amortized approximate maximization," *arXiv:2001.08116*, 2020.

[23] M. V. Pereira and L. M. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical Programming*, vol. 52, no. 1, pp. 359–375, 1991.

[24] J. Dumouchelle, R. Patel, E. B. Khalil, and M. Bodur, "Neur2sp: Neural two-stage stochastic programming," *arXiv:2205.12006*, 2022.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 1st ed. MIT Press, 1998.

[26] S. Fujimoto, D. Meger, D. Precup, O. Nachum, and S. S. Gu, "Why should i trust you, bellman? the bellman error is a poor replacement for value error," *arXiv:2201.12417*, 2022.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[28] IBM, *ILOG CPLEX User's guide*, IBM ILOG, 1987.

[29] C.-F. Daganzo, "An approximate analytic model of many-to-many demand responsive transportation systems," *Transportation Research*, vol. 12, no. 5, pp. 325–333, 1978.

[30] J. Song, Y. Wu, Z. Xu, and X. Lin, "Research on car-following model based on sumo," in *The 7th IEEE/International Conference on Advanced Infocomm Technology*, 2014, pp. 47–55.

[31] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *2015 IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 1–8.

[32] L. Codeca, R. Frank, S. Faye, and T. Engel, "Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017.

[33] C. Ruch, J. Gächter, J. Hakenberg, and E. Frazzoli, "The+ 1 method: model-free adaptive repositioning policies for robotic multi-agent systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3171–3184, 2020.

[34] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of Autonomous Mobility-on-Demand systems," in *Proc. IEEE Conf. on Robotics and Automation*, Stockholm, Sweden, May 2016. [Online]. Available: http://arxiv.org/pdf/1509.03985.pdf

Appendix A
Methodology

In this section, we discuss RL components and network architectures in greater detail.

A. Reinforcement Learning Details

We formulate the AMoD rebalancing problem as an MDP. Specifically, we attempt to learn a behavior policy to select the desired distribution of idle vehicles, as introduced in Section II, by defining the following MDP:

$$\mathcal{M}_{\text{reb}} = (\mathcal{S}_{\text{reb}}, \mathcal{A}_{\text{reb}}, P_{\text{reb}}, r_{\text{reb}}, \gamma). \tag{3}$$

In what follows, we define each of the elements describing the MDP for the AMoD rebalancing problem.

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* $\mathbf{a}^t_{\text{reb}}$. Specifically, the proposed behavior policy will have the task of describing a probability distribution over stations, indicating the percentage of idle vehicles to be rebalanced in each station.

Reward (r_{reb}): We define the reward function in the MDP from the perspective of an AMoD operator. That is, we express our objective 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 at time t will be characterized by a price p^t_{ij} and a cost c^t_{ij}, with $p^t_{ij} = 0$ in case of rebalancing trips. We can naturally express this objective through the following reward function:

$$r_{\text{reb}} = \sum_{i,j \in \mathcal{V}} x^t_{ij}(p^t_{ij} - c^t_{ij}) - \sum_{(i,j) \in \mathcal{E}} y^t_{ij} c^t_{ij}, \tag{4}$$

where, as introduced in Section II, we denote the passenger and rebalancing flows as x^t_{ij} and y^t_{ij}, respectively.

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. 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 $m^t_i \in [0, M], \forall i \in \mathcal{V}$. Given a planning horizon T, we also consider the *projected* availability of idle vehicles $\{m^{t'}_i\}_{t'=t,\dots,t+T}$, where this is estimated based on previously assigned passenger and rebalancing trips. Secondly, an effective rebalancing strategy will also depend on both current d^t_{ij} and estimated $\{\hat{d}^{t'}_{ij}\}_{t'=t,\dots,t+T}$ 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^t_{ij} and cost c^t_{ij}. By means of this definition of the state space, we provide the behavior policy with meaningful information for it to capture statistics of the current and estimated future state of the MoD system, together with operational provider information and performance.

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^t_{ij} is independent of the rebalancing action and follows a time-dependent Poisson process (in our experiments, estimated from real trip travel data). On the other hand, some of the state variable's transitions deterministically depend on the chosen action. For example, the estimated availability $\{m^{t'}_i\}_{t'=t,\dots,t+T}$ is uniquely defined as the sum of the current availability m^t_i 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^t_{ij} and cost c^t_{ij} are assumed to be externally decided and known beforehand (hence, independent from the actions selected by the behavior policy).

B. Neural Network Architecture

Despite our hierarchical policy framework is designed to be agnostic to the choice of neural network architecture, we argue that permutation-invariant computational models operating on irregular graphs—such as those enabled by GNNs—pair naturally with the predominant modeling techniques in transportation engineering broadly and AMoD in particular. At the heart of our claim is the observation that areas in a city, just like nodes in a graph, do not have a natural order. Instead, graphical representations of transportation systems are naturally defined by node and edge properties such as demand in a region or travel time between regions. An effective control policy should not be affected by the order in which we consider the areas, but rather solely by the properties (i.e. attributes) of those areas. We first introduce the basic building blocks of our graph neural network architecture.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}_{i=1:N_v}$ and $\mathcal{E} = \{e_k\}_{k=1:N_e}$ respectively define the sets of nodes and edges of $\mathcal{G}$, most current graph neural network models can be seen as methods attempting to learn a permutation-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. An architecture of particular interest for this work is the Graph Convolution Network (GCN) [20]. At its core, a graph convolutional operator describes a parametric function $f(\mathbf{X}, \mathbf{A})$ for efficient information propagation on graphs. Specifically, a GCN defines the following propagation rule:

$$\mathbf{X}' = f(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}\right), \tag{5}$$

where $\mathbf{X}$ is the $N_v \times D_{\mathbf{x}}$ feature matrix, $\mathbf{A}$ is the adjacency matrix with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\mathbf{I}$ is the identity matrix. $\hat{\mathbf{D}}$ is the diagonal node degree matrix of $\hat{\mathbf{A}}$, $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU) and $\mathbf{W}$ is a matrix of learnable parameters.

Policy. As introduced in Section II, 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. Concretely, the neural network used in our implementation consists of one layer of graph convolution with skip-connections and ReLU activations, whose output is then aggregated across neighboring nodes using a permutation-invariant sum-pooling function, and finally passed to three MLP layers of 32 hidden units to produce the Dirichlet concentration parameters.

Value Function. The architecture used to define the value function $V_\phi(\mathbf{s}_t)$ is in many ways identical to the architecture used to characterize the policy. The main difference between the two architectures lies in an additional *global* sum-pooling performed on the output of the graph convolution. In this way, the value function is able to aggregate information across all nodes in the graph, thus computing a single value function estimate for the entire network.

APPENDIX B
DISCUSSION AND ALGORITHMIC COMPONENTS

In this section, we discuss various elements of the proposed framework, highlight correspondences and design decisions, and discuss component-level extensions.

A. Computational efficiency

Consider solving the full nonlinear network flow problem via direct optimization over a finite horizon (T timesteps), which corresponds to a model predictive control [21] formulation. How many actions must be selected? The number of possible flows for a fully dense graph (worst case) is $N_v(N_v - 1)$. Thus, the worst-case number of actions to select is $TN_v(N_v - 1)$; it is evident that for even moderate choices of each variable, the complexity of action selection in our problem formulation quickly grows beyond tractability.

While moderately-sized problems may be tractable within the direct optimization setting, we aim to incorporate the impacts of stochasticity, nonlinearity, and uncertainty, which typically result in non-convexity. The reinforcement learning approach, in addition to being able to improve directly from data, reduces the number of actions required to those for a single step. If we were to directly parameterize the naive policy that outputs flows, this would correspond to $N_v(N_v-1)$ actions. For even moderate values of N_v, this can result in millions of actions. It is well-known that reinforcement learning algorithms struggle with high dimensional action spaces [22], and thus this approach is unlikely to be successful. In contrast, our hierarchical formulation requires only N_v actions for the learned policy, while additionally leveraging the beneficial inductive biases over short time horizons.

B. Goal-reaching constraint as value function

In our framework, the low-level minimum rebalancing cost step is designed to minimize the distance between the *realized* and *desired* vehicle distribution (i.e., obtained as output from the RL policy). The role of this distance metric (and the generated desired vehicle distribution) is to capture the value of future reward in the greedy one-step inner optimization problem. This is closely related to the value function in dynamic programming and reinforcement learning, which in expectation captures the sum of future rewards for a particular policy. Indeed, under moderate technical assumptions, our linear problem formulation with stochasticity yields convex expected cost-to-go (the negative of the value) [23, 24].

There are several critical differences between our penalty term and a learned value function. First, a value function in a Markovian setting for a given policy is a function solely of state. For example, in the minimum rebalancing cost step, a value function would depend only on $\mathbf{s}_{t+1}$. In contrast, our value function depends on $\mathbf{a}_t$, which is the output of a policy that takes $\mathbf{s}_t$ as an input. Thus, the penalty term is a function of both the current and desired vehicle distribution. Given this, the penalty term is better understood as a local approximation of the value function, for which convex optimization is tractable, or as a form of state-action value function with a reduced action space (also referred to as a Q function).

The second major distinction between the penalty term and a value function is particular to reinforcement learning. Value functions in modern RL are typically learned via minimizing the Bellman residual [25], although there is disagreement on whether this is a desirable objective [26]. In contrast, our policy is trained directly via gradient descent on the total reward (potentially incorporating value function control variates). Thus, the objective of this penalty method is better aligned with maximizing total reward.

Appendix C
Additional Experimental Details

In this section, we provide additional details of the experimental setup and hyperparameters. All RL modules were implemented using PyTorch [27] and the IBM CPLEX solver [28] for the optimization problem.

A. Environment details

Macroscopic simulator. We use two case studies from the cities of New York, USA, and Chengdu, China, whereby we study a hypothetical deployment of taxi-like systems to serve the peak-time commute demand in popular areas of Brooklyn and Chengdu, respectively. The cities are divided into geographical areas, each of which represents a station. The case studies in our experiments are generated using trip record datasets, which we provide together with our codebase. The trip records are converted to demand, travel times, and trip prices between stations. Here, we consider stochastic time-varying demand patterns, whereby customer arrival is assumed to be a time-dependent Poisson process, and the Poisson rates are aggregated from the trip record data every 3 minutes. We assume the stations to be spatially connected, whereby moving vehicles from one station to the other requires non-trivial sequential actions (i.e., vehicles cannot directly be repositioned from one station to any other station, rather they have to adhere to the available paths given by the city's topology).

The following remarks are made in order. First, we assume travel times are given and independent of operator actions. This assumption applies to cities where the number of vehicles in the fleet constitutes a relatively small proportion of the entire vehicle population on the transportation network, and thus the impact on traffic congestion is marginal. This assumption can be relaxed by training the proposed RL model in an environment considering the endogenous congestion caused by controlled vehicle fleets. Second, without loss of generality, we assume that the arrival process of passengers for each origin-destination pair is a time-dependent Poisson process. We further assume that such a process is independent of the arrival processes of other origin-destination pairs and the coordination of vehicles. These assumptions are commonly used to model transportation requests [29].

Mesoscopic simulator. The mesoscopic model offers a hybrid approach to traffic simulation that merges the intricate vehicle-level behaviors from microscopic models with the aggregated traffic flow characteristics of macroscopic models. Microscopic models commonly leverage a car-following approach, such as the Krauss model [30], providing a granular representation of traffic dynamics at the single vehicle level. Such detailed simulations, however, require substantial computational resources. Notably, when simulating a scenario for an AMoD operator, the required level of accuracy is typically lower, and one can model car-following behaviors at a coarser level, reducing the computational burden. To this end, mesoscopic models employ queuing theory to spatially aggregate the car flow description [18]. Each link of a network is represented as a sequence of queues, in which each car entering the queue has to wait at least the free-flow travel time before leaving it,

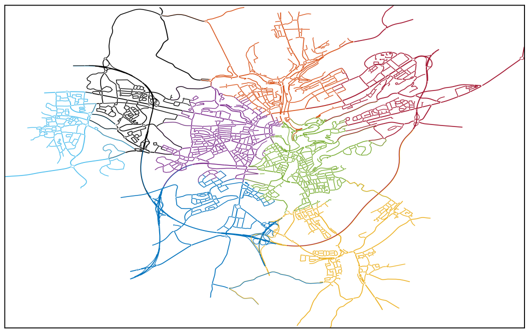

Fig. 4. Aggregated view of the Luxembourg city network.

calculated as follows:

$$t^{\nu}_{tr} = L^i / v^i_{max}, \tag{6}$$

where L^i represents the length of the i-th queue, and v^i_{max} denotes its free-flow speed. In case of a congested queue, the waiting time depends on the number of vehicles in the queue. This can be defined to link the exit time of vehicle ν with the exit time of the preceding vehicle $\nu - 1$ as follows:

$$t^{\nu}_{exit} \geq t^{\nu-1}_{exit} + \tau^i_s, \tag{7}$$

where the time headway associated with the i-th queue, τ^i_s, depends on the congestion level of the queue. To differentiate between free-flow and congested queues, a congestion parameter n^i_{jam} must be defined. This parameter allows for the determination of four distinct values for the time headway, based on the congestion state of both the current queue and the subsequent queue, as follows:

$$\tau^i_s = \begin{cases} \tau_{ff} & \text{if } n^i < n^i_{\text{jam}} \text{ and } n^{i+1} < n^{i+1}_{\text{jam}} \\ \tau_{fj} & \text{if } n^i < n^i_{\text{jam}} \text{ and } n^{i+1} \geq n^{i+1}_{\text{jam}} \\ \tau_{jf} & \text{if } n^i \geq n^i_{\text{jam}} \text{ and } n^{i+1} < n^{i+1}_{\text{jam}} \\ f(n^{i+1}, \tau_{jj}) & \text{if } n^i \geq n^i_{\text{jam}} \text{ and } n^{i+1} \geq n^{i+1}_{\text{jam}}, \end{cases} \tag{8}$$

where n^i is the number of vehicles in the i-th queue. The high-fidelity scenario described in Section III-D is based on a mesoscopic model of the city of Luxembourg. Detailed information on the calibration of this traffic scenario can be found in [31, 32]. In this scenario, the AMoD operator is modeled as a robo-taxi fleet that must serve travel demand. Additionally, exogenous traffic and road infrastructure elements such as stop signs, lanes, and traffic lights—are also considered. Such factors directly influence the network's congestion levels and, consequently, the travel time, resulting in a more realistic depiction of traffic flow behavior when compared to macroscopic models. The original network has been aggregated into 8 different regions, as shown in Figure 4. The desired vehicle distribution will then be determined for these 8 aggregated regions. Additionally, each passenger can request a ride only between the centers of these regions. The same assumption applies when the vehicles are rebalanced.

TABLE IV
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
RL-0Shot	46,516 (**-15.1%**)	13,974	10,083
RL	47,843 (**-12.6%**)	14,165	12,165
MPC-oracle	54,737 (0%)	16,275	10,389

B. Model implementation

In what follows, we provide additional details for the implemented baselines and models:

Domain-driven heuristics. Within this class of methods, we measure the performance of heuristics generally accepted as reasonable baselines.

1) *Equally-balanced System*: at each decision step, we take rebalancing actions so to recover an equal distribution of idle vehicles across all areas in the transportation network. Concretely, the heuristic achieves this by solving the minimum rebalancing cost problem with a fixed desired number of idle vehicles among all stations, i.e., given M available vehicles at time t, $\hat{m}^{t+1} = \{\frac{M}{|\mathcal{N}|}\}_{i \in \mathcal{N}}$.
2) *Plus-1*: at each decision step, we recover the number of vehicles available in the region proceeding with the demand-matching process. For every vehicle that departs a region to fulfill a passenger trip, another vehicle is rebalanced back to that same region. This method has been presented in [33].

Optimization-based methods. Within this class of methods, we measure the performance of methods relying on the solution of large-scale network flow problems.

3) *MPC-Oracle*: we directly optimize the passenger flow and rebalancing flow using a standard formulation of MPC [34] that assumes perfect foresight information of future user requests and network conditions (e.g., travel times, prices, etc.). Therefore, this approach serves as an *oracle* that provides a performance upper bound for any coordination algorithm.
4) *MPC-Forecast*: we relax the assumption of perfect foresight information in MPC-Oracle, substituting with a noisy and unbiased estimate of demand. This estimate takes the form of the rate of the underlying time-dependent Poisson process describing passenger arrival in the system. This approach is a realistic control-based benchmark in the context of unknown system dynamics but also exhibits poor scalability like MPC-Oracle.

C. Additional results

Service Area Expansion. To further study how well the RL-based policy can generalize to conditions unseen during training, we now consider the case of a hypothetical service area expansion. We define the task as follows: given 64 stations in the New York scenario (organized as a 8×8 grid over the city's geography), we first learn a rebalancing policy in the inner 4×4 region, which we then use, without any fine-tuning, on the full 8×8 grid. Despite the 300% increase in service area, results in Table IV show that RL-0Shot is only 2.5% less profitable and satisfies 1.3% fewer customers

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

	Reward (%Dev. MPC-oracle)	Served Demand	Rebalancing Cost ($)
ED	7,900 (-27.9%)	2,431	3,451
RL	9,981 (**-8.9%**)	2,531	1,371
MPC-oracle	10,955 (0%)	2,527	382

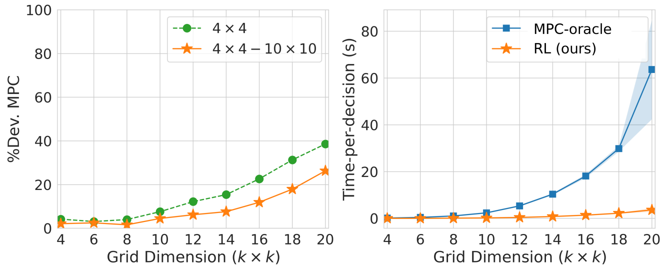

Fig. 5. Left: System performance (Percentage Deviation from MPC-oracle) for agents trained either on a single granularity (4×4) or across granularities (4×4 - 10×10), Right: Comparison of computation times between RL and MPC-oracle.

when compared to its fully-retrained counterpart (RL), thus exhibiting an interesting degree of portability to scenarios unseen during training. From a practical perspective, it is important to underline how this experimental setting might represent a set of common real-world scenarios, such as: (i) a service provider interested in expanding its service area without having to re-train a control policy from scratch, and (ii) when faced with extremely large urban networks, a service provider might consider training a policy on specific subgraphs of the network, and later being able to deploy the learned policy on the entire system.

Irregular Geographies.

We now investigate how well the pre-trained RL 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 over the city's geography. Results in Table V show that the proposed approach achieves almost 40% reduction in rebalancing cost together with a 26% increase in profit compared to an equally distributed policy, thus exhibiting natural adaptation capabilities to irregular geographies.

Most importantly, the permutation of areas and their disjoint positioning, would make non-GNN-based architectures ill-defined for the task. On the other hand, by explicitly representing stations as relational entities in a graph, graph neural networks enable reinforcement learning agents to recover extremely flexible behavior policies when dealing with diverse and irregular urban topologies.

Network Granularity.

From a service provider perspective, given a spatial segmentation of the service area, a critical decision is the one characterizing the spatial area associated with each node, or *granularity*, of each rebalancing area. To assess how well RL is able to generalize across different service granularities we also study adaptation to finer spatial segmentations. Specifically, we define the task as follows: given arbitrary granularities ranging from 4×4 to 20×20 grids (with 2×2 increments), we are interested in evaluating the portability of RL when trained on coarse granularities and later applied, without any fine-tuning, to finer spatial scales. Results in Fig. 5 (left) show transfer performance for RL under two different training strategies. The first pre-trains a rebalancing policy solely on a 4×4 grid, while the second exposes the agent to a diversity of granularities also at training time by considering grids up until 10×10. The results show that training across granularities strongly increases the generalization capabilities of RL, leading to rebalancing policies effective also under extreme granularity variations. Crucially, we believe these results show clear evidence that it is possible to explicitly consider transfer and generalization in the design of the networks and training pipelines, and we believe this is an interesting and fruitful direction for future work.