2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Learning to Control Autonomous Fleets from Observation via Offline Reinforcement Learning

Carolin Schmidt[1], Daniele Gammelli[2], Francisco Camara Pereira[1], and Filipe Rodrigues[1]

***Abstract*— Autonomous Mobility-on-Demand (AMoD) systems are an evolving mode of transportation in which a centrally coordinated fleet of self-driving vehicles dynamically serves travel requests. The control of these systems is typically formulated as a large network optimization problem, and reinforcement learning (RL) has recently emerged as a promising approach to solve the open challenges in this space. Recent centralized RL approaches focus on learning from online data, ignoring the per-sample-cost of interactions within real-world transportation systems. To address these limitations, we propose to formalize the control of AMoD systems through the lens of offline reinforcement learning and learn effective control strategies using solely offline data, which is readily available to current mobility operators. We further investigate design decisions and provide empirical evidence based on data from real-world mobility systems showing how offline learning allows to recover AMoD control policies that (i) exhibit performance on par with online methods, (ii) allow for sample-efficient online fine-tuning and (iii) eliminate the need for complex simulation environments. Crucially, this paper demonstrates that offline RL is a promising paradigm for the application of RL-based solutions within economically-critical systems, such as mobility systems.**

I. INTRODUCTION

As the world's population continues to urbanize, with projections indicating that over 60% of the population will reside in urban environments by 2050, there is an urgent need for innovative mobility solutions [1]. The traditional model of urban transportation, heavily reliant on private cars, is no longer sustainable and is widely recognized as the main cause of increasingly congested transportation systems. Within this context, Autonomous Mobility-on-Demand (AMoD) systems have the potential to transform urban transportation by providing cheap and efficient point-to-point trips while reducing the need for private car ownership. In an AMoD system, the customer requests a one-way ride from their origin to a destination and is matched with an autonomous vehicle belonging to a larger fleet. In real-world systems, the effectiveness of rebalancing strategies is central to the overall system performance, with sub-optimal strategies potentially exacerbating congestion through unnecessary trips or increased passenger waiting times. In practice, AMoD systems enable the centralized control of the fleet, thus potentially allowing for extremely efficient transportation systems. However, the task of managing and routing a large number of vehicles within real-world transportation systems is typically formulated as a large network optimization problem, which may be prohibitively expensive in practice, and its solution is considered an open problem.

[1]Carolin Schmidt, Francisco Camara Pereira and Filipe Rodrigues are with the Department of Technology, Management and Economics, Technical University of Denmark, Kongens Lyngby, Denmark {csasc, camara, rodr}@dtu.dk

[2]Daniele Gammelli is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, USA, gammelli@stanford.edu

Recently, reinforcement learning (RL) has emerged as a promising approach to solving the AMoD control problem while avoiding expensive optimization routines [2], [3], [4], [5]. A traditional approach to RL typically entails the definition of an agent that gradually improves its performance by repeatedly interacting with an environment. However, the necessity to learn from online data is also one of the major constraints to its real-world adoption, especially within safety or economically-critical systems [6] (e.g., healthcare, autonomous driving, etc.). An alternative approach is to train via simulation, which can be summarized as (i) building a complex simulator of urban mobility, (ii) training AMoD controllers in simulation, and (iii) deploying the agent in the real world [7]. However, these approaches require access to reliable simulators – in itself a challenging task – and are potentially exposed to sim-to-real distribution shifts [8]. In this work, we argue that offline RL represents a promising direction to overcome these challenges by learning from real, pre-collected data. Crucially, online interaction within real-world AMoD systems is either extremely expensive or infeasible, while historical data from service operators is likely to be abundant and readily available. Thus, we propose offline RL as a framework to enable service operators to completely avoid expensive fleet management decisions until a sufficient level of performance is guaranteed, allowing for further online fine-tuning once the policy is safer and cheaper to deploy and taking a step toward the deployment of learning-based solutions for the system-level control of real-world transportation systems.

The contributions of this paper are threefold:

- We formulate the AMoD control problem through the lens of offline RL and propose an approach that enables RL agents to centrally control AMoD systems from solely offline data.
- We investigate design decisions within our framework, such as the relation between dataset quality and coverage, and quantify their impact on model performance and transferability.
- We show that our approach learns policy initializations that enable sample-efficient online fine-tuning, providing a practically feasible strategy to deploy RL algorithms within real-world mobility systems.

II. RELATED WORK

Existing literature on the AMoD control problem can be broadly classified into rule-based heuristics [9], model predictive control (MPC) methods [10], and RL approaches. Readers interested in a recent survey about centralized and decentralized RL approaches for vehicle relocation can refer to [11].

As AMoD systems enable the centralized control of the fleet, we focus on reviewing system-level vehicle rebalancing with central decision-making. Such methods aim to explicitly modify the current distribution of idle vehicles to collectively improve system performance and serve more requests. Due to the intractability of the joint action space for all vehicles, centralized RL approaches for the control of ride-hailing systems have only been studied recently. To tackle this challenge of scalability, [2] devise a system policy that specifies the aggregated number of vehicles to be rebalanced between zones, [3] formulate a policy that determines a sequence of atomic actions, and [12] deploy a cascaded multi-level Q-learning approach. [13] train a policy that specifies a probability distribution for rebalancing idle vehicles to their k-nearest neighbors. [4] propose a graph neural network-based Dirichlet policy that determines the desired idle vehicle distribution in the network.

Overall, approaches based on RL demonstrate computational tractability without significantly sacrificing optimality. However, despite their effectiveness, the aforementioned studies do not consider the cost of training an RL agent from scratch, which can be extremely high within real-world systems (i.e., consider the monetary cost of trying different fleet rebalancing strategies for the purpose of RL exploration). No publicly known simulation environment for ride-hailing services has demonstrated sufficient fidelity to the real world for direct sim-to-real transfer [11]. Thus, deployed work has either adopted offline [14], [15] or fully online learning of value functions [16], [17]. To address the challenges of the high-dimensionality state space, these studies assume that the global value function can be decomposed into the individual drivers' value functions. Such an assumption is valid where individual goals align with the global system's objective, e.g., in vehicle dispatching. However, for vehicle rebalancing, this assumption does not hold and would lead to idle drivers clustering at a single high-value location. To mitigate this undesirable behavior, [14] and [15] sample a destination for each driver from a Boltzmann distribution. [16], [17] only consider vehicle dispatching, but their approach could be extended accordingly. This method avoids the need for an explicit policy but may compromise system performance due to limited coordination. From the AMoD service provider's perspective, the elimination of drivers renders individual objectives obsolete, placing the focus on system-wide objectives for successful operations. A solution lies in a centralized parametric policy, which aligns vehicle actions with system goals by explicitly optimizing for system performance as in [2], [3], [12], [4], [13]. However, if aiming at deploying a parametric policy, fully online approaches [16], [17] could not be deployed, as an untrained policy could incur significant real-world costs when interacting with the AMoD system. In light of this, learning a policy from an offline dataset requires explicit offline RL methods [6], while previous offline approaches [14], [15] focus solely on deriving value estimates from the dataset. In this work, we aim to address these challenges by proposing offline RL to learn a policy for system-level rebalancing as a mathematical framework to eliminate the need for online learning from scratch and learning in simulation environments.

III. BACKGROUND

In this section, we introduce the notation and theoretical background underlying our work in the context of RL, offline RL, conservative Q-learning, and online fine-tuning.

A. The Reinforcement Learning Problem

In reinforcement learning, we aim to learn to control a dynamic system from experience. Specifically, we consider a (fully-observable) infinite-horizon Markov decision process (MDP) $M=(S,A,P,d_0,r,\gamma)$, where S is a set of possible states $s\in S$, A is a set of possible actions $a\in A$, P defines the conditional probability distribution $P(s_{t+1}|s_t,a_t)$ that specifies the dynamics of the system, d_0 defines the initial state distribution $d_0(s_0)$, $r: S\times A\rightarrow\mathbb{R}$ describes a reward function and $\gamma\in(0,1]$ is a scalar discount factor. The goal of an RL agent is to learn a policy, which defines a distribution over possible actions conditioned on the state $\pi(a_t|s_t)$ by interacting with the MDP M (i.e., the environment) under the objective of maximizing the expected sum of cumulative rewards. An approach of particular interest for this work is the Soft Actor-Critic (SAC) [18] algorithm, which combines an actor-critic formulation with maximum entropy reinforcement learning. We consider a parametric policy $\pi_\phi(a_t|s_t)$, i.e., the "actor", that maps states to actions, and a Q-function $Q_\theta(s_t,a_t)$, i.e., the "critic", that estimates the expected return when starting from state s, taking an arbitrary action a, and then continuing acting according to policy π. The Q-value is optimized to approximate the expected return under the current policy π. Specifically, the Q-values are updated by iteratively applying the soft Bellman backup operator B^π given by:

$$B^\pi Q(s_t,a_t)=r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim P}[\mathbb{E}_{a_{t+1}\sim\pi}[Q(s_{t+1},a_{t+1})-\alpha\log(\pi_\phi(a_{t+1}\,|\,s_{t+1}))]],$$

where (s_t,a_t) is sampled from a collection of past transitions $\mathscr{D}=\{(s_t,a_t,s_{t+1},r_t)\}$, often referred to as a replay buffer. The policy is updated to maximize the expected return, represented by the Q-values, alongside the entropy of the policy:

$$J_\pi(\phi)=\mathbb{E}_{s_t\sim\mathscr{D}}\Big[\mathbb{E}_{a_t\sim\pi_\phi}\big[\alpha\log\big(\pi_\phi(a_t\,|\,s_t)\big)-Q_\theta(s_t,a_t)\big]\Big].$$

B. Offline Reinforcement Learning

Offline RL is a learning paradigm that involves learning on a static dataset of transitions i $\mathscr{D}=\{(s_t^i,a_t^i,s_{t+1}^i,r_t^i)\}$ collected by a potentially sub-optimal policy π_β. The fundamental challenge of offline RL is to find a strategy that is different from the behavior policy π_β and, at the same time, avoids erroneous behavior outside the data distribution. For a comprehensive review of offline RL approaches, interested readers can refer to [6], [19].

In this work, we exploit ideas from the conservative Q-learning algorithm (CQL) [20]. In Q-learning, querying the value function on out-of-distribution actions that are not forced to obey the Bellman residual typically leads to an overestimation of Q-values. As introduced in the CQL algorithm, this overestimation can be solved via regularization of the Q-function, concretely incentivizing a conservative estimate of the true Q-function by minimizing Q-values in addition to the standard Bellman error. Formally, the training objective of CQL, in addition to the standard bellman residual, is given by:

$$\min_Q \eta\mathbb{E}_{s\sim\mathscr{D},a\sim\pi}[Q(s,a)]-\mathbb{E}_{s,a\sim\mathscr{D}}[Q(s,a)],$$

where η is the regularizer weight that controls the trade-off between the Bellman residual and the CQL regularization term. This formulation has proven to be an effective and competitive way of training an agent offline [20], [21].

With offline RL, we obtain a policy initialization, which is intended for sample-efficient online fine-tuning. However, conservative methods tend to learn smaller Q-values than their true values. Consequently, initial interactions during online fine-tuning are spent adjusting the Q-function, leading to unintentional unlearning of the initial policy. To address this, [22] propose calibrating the Q-function during offline learning to match the range of the ground-truth Q-values. Their approach Cal-CQL ensures the learned, conservative Q-values are lower-bounded by the ground-truth Q-values of a sub-optimal reference policy, denoted as μ, for which we can choose the behavior policy of the dataset. Practically, this is achieved by masking out the push-down on the Q-value for out-of-distribution (OOD) actions if they are not calibrated. This modifies the CQL regularizer to:

$$\min_{Q} \eta \mathbb{E}_{s\sim\mathscr{D},a\sim\pi}[max(Q(s,a),V^{\mu}(s))]-\mathbb{E}_{s,a\sim\mathscr{D}}[Q(s,a)].$$

IV. The AMoD Problem

We now define the terminology associated with the AMoD control problem. An on-demand service provider coordinates M single-occupancy autonomous vehicles on a transportation network represented by a complete graph $\mathscr{G}=(\mathscr{V},\mathscr{E})$ where $\mathscr{V}=\{v_i\}_{\{i=1:N_v\}}$ and $\mathscr{E}=\{e_j\}_{\{j=1:Ne\}}$ represent the set of vertices and edges of $\mathscr{G}$. Specifically, $\mathscr{V}$ defines the set of stations (e.g., pick-up or drop-off locations), and $\mathscr{E}$ defines the shortest paths between stations. The time horizon is discretized into a set of time steps $\mathscr{I}=1,2,...,T$ of length T. At any time step t, vehicles are controlled to travel along the shortest path between station i and $j\neq i\in\mathscr{V}$ with a travel time of $\tau^t_{i,j}\in\mathbb{Z}^+$ and travel cost c_{ij}, as a function of travel time. At each time step t, passengers submit trip requests for a desired origin-destination pair $(i,j)\in\mathscr{V}\times\mathscr{V}$, which is characterized by demand $d^t_{i,j}$ and price $p^t_{i,j}$. The operator matches passengers to vehicles, and the vehicles will transport the passengers to their destinations. For idle vehicles that are not matched with any passengers, the operator controls them to stay at the same station or rebalance to other stations. We denote $x^t_{i,j}\in\mathbb{N},x^t_{i,j}\leq d^t_{i,j}$ as the passenger flow, i.e., the number of passengers traveling from station i to station j at time t and $y^t_{i,j}\in\mathbb{N}$ as the rebalancing flow, i.e., the number of vehicles rebalancing from station i to station j at time t.

V. Offline Reinforcement Learning for AMoD

In this section, we introduce a three-step optimization framework for effective AMOD fleet management before we formulate the AMoD rebalancing MDP and formalize Conservative Q-learning for AMoD systems.

A. The Three-step Framework

As in [4], [5], [23], this paper adopts a three-step decision-making framework to tackle the AMoD control problem. This framework comprises three stages: (1) solving a matching problem to derive the passenger flow, (2) determining the desired distribution of idle vehicles through the use of the learned policy $\pi_\phi(a_t|s_t)$, (3) converting this distribution to a rebalancing flow by solving a minimal rebalancing-cost problem. The advantage of this procedure is the reduction of the action space from $|\mathscr{V}|^2$ to $|\mathscr{V}|$, as the policy defines the action at each node as opposed to along each edge (i.e., each origin-destination pair).

In the first step, vehicles are assigned to customers by solving the following assignment problem to derive passenger flows $\{x^t_{i,j}\}_{i,j\in\mathscr{V}}$:

$$\max_{\{x^t_{ij}\}_{i,j\in\mathscr{V}}} \sum_{i,j\in\mathscr{V}} x^t_{ij}(p^t_{ij}-c^t_{ij}) \tag{1a}$$

$$\text{s.t.} \quad 0\leq x^t_{ij}\leq d^t_{ij},\ i,j\in\mathscr{V}, \tag{1b}$$

$$\sum_{j\in\mathscr{V}} x^t_{ij}\leq M^t_i,\ i\in\mathscr{V}, \tag{1c}$$

where objective function (1a) maximizes the total profit, which is computed as the difference between revenue and the cost of the matched trips. Constraint (1b) ensures that the resulting passenger flow is non-negative and upper-bounded by the demand, while constraint (1c) guarantees that the assigned passenger flow does not exceed the number of available vehicles M^t_i at station i at time step t. Note that since the constraint matrix is totally unimodular, the resulting passenger flows are integers as long as the demand is integral.

Within the second step, the learned policy $\pi_\phi(a_t|s_t)$ determines the desired idle vehicle distribution $a^t_{reb}=\{a^t_{reb,i}\}_{i\in\mathscr{V}}$, where $a^t_{reb,i}\in[0,1]$ defines the percentage of currently idle vehicles to be rebalanced towards station i in time step t, and $\sum_{i\in\mathscr{V}}a^t_{reb,i}=1$. Given the desired vehicle distribution, we denote the number of desired vehicles as $\hat{m}^t_i=\lfloor a^t_{reb,i}\sum_{i\in\mathscr{V}}m^t_i\rfloor$, where m^t_i represents the actual number of idle vehicles in region i at time step t.

The third step converts the desired distribution into rebalancing flows $\{y^t_{i,j}\}_{i,j\in\mathscr{E}}$ by solving a minimal rebalancing cost problem:

$$\min_{\{y^t_{ij}\}_{i\neq j\in\mathscr{V}}\in\mathbb{N}^{|\mathscr{V}|\times(|\mathscr{V}|-1)}} \sum_{i\neq j\in\mathscr{V}} c^t_{ij}y^t_{ij} \tag{2a}$$

$$\text{s.t.} \quad \sum_{j\neq i}(y^t_{ji}-y^t_{ij})+m^t_i\geq\hat{m}^t_i,\ i\in\mathscr{V}, \tag{2b}$$

$$\sum_{j\neq i} y^t_{ij}\leq m^t_i,\ i\in\mathscr{V}, \tag{2c}$$

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

B. The AMoD Rebalancing MDP

Our goal is to learn a policy to compute the desired distribution of idle vehicles (Step 2) via offline data. However, we want to emphasize that our approach is agnostic to the framework and can be seen as an indication for other approaches that learn rebalancing policies. We define the AMoD rebalancing problem as an MDP $M^{reb}=(S^{reb},A^{reb},P^{reb},d^{reb}_0,r^{reb},\gamma)$ characterized as following:

Reward (r^{reb}): we choose the reward to be the operator's profit, which we define as the difference between the revenue from serving passengers and the cost of operations:

$$r^t_{reb}=\sum_{i,j\in\mathscr{V}} x^{t+1}_{ij}(p^{t+1}_{ij}-c^{t+1}_{ij})-\sum_{(i,j)\in\mathscr{E}} y^t_{ij}c^t_{ij}.$$

Action space (A^{reb}): given the number of idle vehicles and their current spatial distribution, we consider the problem of determining the *desired idle vehicle distribution* a^t_{reb}. Specifically, the policy

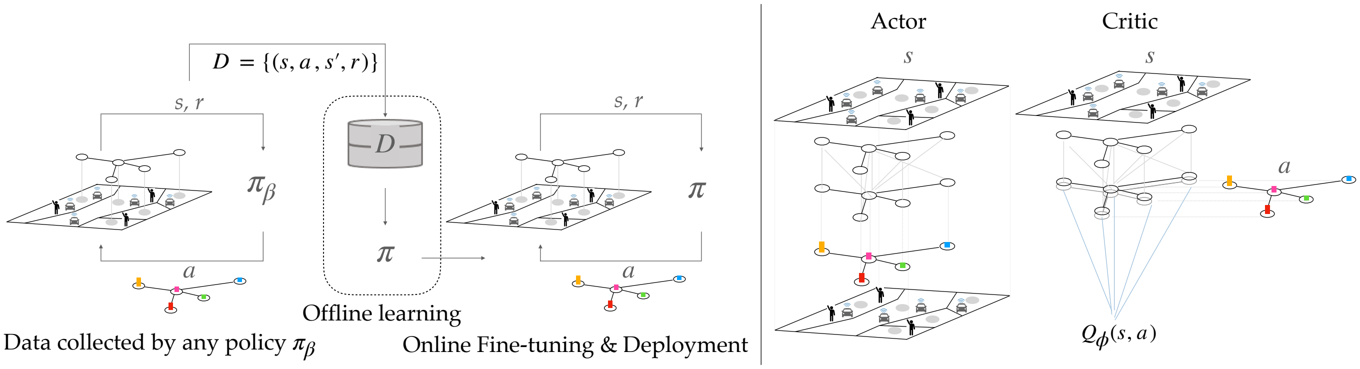

Fig. 1. (Left) The policy π is trained on a dataset collected by a (potentially sub-optimal) policy π_β without any environment interaction before being deployed. (Right) Both actor and critic update raw graph representations of the transportation network to compute (i) a desired distribution of idle vehicles a and (ii) an estimate of the Q-function, respectively.

describes a probability distribution over stations, indicating the percentage of idle vehicles to be rebalanced towards each station.

State space (S^{reb}): we define the state to contain the information needed to determine proactive rebalancing strategies, including the structure of the transportation network through its adjacency matrix $\mathbf{A}$ and state-level information by means of a feature matrix $\mathbf{X}$. Specifically, given a planning horizon K, we consider: (1) the current and projected availability of idle vehicles in each station $m_i^t \in [0,M], \forall i \in \mathcal{V}$ and $\{m_{i,j}^{t'}\}_{t'=t,\dots,t+K}$, (2) provider-level information trip price $p_{i,j}^t$ and cost $c_{i,j}^t$, (3) current d_{ij}^t and estimated $\{\hat{d}_{i,j}^{t'}\}_{t'=t,\dots,t+K}$ transportation demand between all stations. 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 prediction model.

Dynamics (P^{reb}): The stochastic evolution of travel demand between stations follows a time-dependent Poisson process, with a time-varying arrival rate estimated from real data. The process is independent of the rebalancing action and the arrival process of other passengers in other locations. Additionally, the evolution of state elements is characterized as follows: (1) the estimated availability $\{m_{i,j}^{t'}\}_{t'=t,\dots,t+K}$ is determined based on the number of idle vehicles at station i at time step $t+1$ plus the number of incoming vehicles minus the number of outgoing vehicles (from both passenger and rebalancing flow), (2) provider-level state variables, such as trip price p_{ij}^t and cost c_{ij}^t, are assumed to be exogenous and known beforehand.

C. Conservative Q-Learning for AMoD Systems

We propose a SAC formulation with the conservative loss of CQL as an approach to learning rebalancing policies offline. We assume the availability of a dataset $\mathscr{D} = \{(s_t, a_t, s_{t+1}, r_t)\}$ containing historical rebalancing decisions collected by a behavior policy π_β. Our objective is to train an agent using this dataset without any further interactions with the AMoD system and to achieve a policy that matches or even outperforms π_β. The offline learning process and the actor and critic architectures are illustrated in Figure 1. We use GNN encoders to parameterize the neural architecture for the policy $\pi_\phi(a_t|s_t)$ and the Q-function $Q_\theta(s_t, a_t)$. GNNs have proven to be particularly effective in transportation applications due to their ability to capture and process spatial relations within the network. Additionally, GNNs can process transportation networks of varying sizes and connectivity, while traditional machine learning methods, such as MLPs, are limited to fixed-size inputs. In what follows, we introduce the neural network architectures in more detail:

Policy network $\pi_\phi(a|s)$. To define a valid vehicle distribution, the output of the policy network is sampled from a Dirichlet distribution. More precisely, the network architecture comprises one graph convolutional layer with skip-connections that uses a sum-pooling function as neighborhood aggregation and a ReLU non-linearity on its output. This is followed by three MLP layers (with ReLU activations on the first two) that output the concentration parameter c for the Dirichlet distribution. To ensure the positivity of c, we apply a softplus activation function in the last layer.

Critic network $Q_\theta(s_t,a_t)$. We propose an architecture for the critic which uses the same GNN encoder as the policy network. The main difference to the actor architecture is that the encoded state information is concatenated with the action on a node level, followed by two MLP layers with ReLU activation. The global sum-pooling function is not performed until before the last layer. We found this architecture choice to be essential for satisfactory results.

VI. Experiments

In this section, we present simulation results using data from different cities. Specifically, the goal of our experiments is to answer the following questions: (1) can a policy solely trained on a static dataset learn effective rebalancing strategies in real-world urban mobility scenarios? (2) how do characteristics of the dataset influence the performance and transfer capabilities of offline RL agents? and (3) can we quantify the benefits of deploying an agent trained offline compared to one trained from scratch via online data? [1]

A. Benchmarks

In our experiments, we compare our proposed offline-RL framework with the following methods:

[1](Code, data and hyperparameters available: `https://github.com/carolinssc/offline-rl-amod`)

TABLE I
AVERAGE REWARD (PROFIT, THOUSANDS OF DOLLARS). BOLD HIGHLIGHTS BEST PERFORMING (NON-ORACLE) MODEL.

City	Random	No Reb.	ED	CQL	SAC	MPC-Forecast	MPC-Oracle
NYC Brooklyn	26.5($\pm$0.4)	27.6($\pm$0.6)	49.2($\pm$0.9)	53.1($\pm$1.1)	56.1($\pm$1.1)	**56.5**($\pm$1.2)	57.2($\pm$0.8)
Shenzhen West	48.6($\pm$1.7)	60.0($\pm$0.7)	58.4($\pm$0.9)	61.6($\pm$0.5)	**62.3**($\pm$0.7)	60.6($\pm$1.0)	65.2($\pm$1.0)
San Francisco	12.5($\pm$0.6)	10.1($\pm$0.4)	14.1($\pm$0.4)	14.2($\pm$0.4)	**14.8**($\pm$0.3)	13.9($\pm$0.5)	15.9($\pm$0.4)

No rebalancing: we measure the performance of the system without any rebalancing activities.

Heuristics:

1) *Random policy*: at each time step, the desired distribution is sampled from a Dirichlet prior with a concentration parameter $c=[1,1,...,1]$.
2) *Equally distributed policy* (ED): rebalancing actions are selected to restore an equal distribution of vehicles across all areas in the transportation network.

Learning-based: within this class of methods, we measure the performance of RL agents trained via online learning.

1) *SAC*: the online version of our method based on the Soft Actor-Critic algorithm. In practice, this benchmark serves as an upper bound of performance for our offline approach as we define both actor and critic networks as in our CQL formulation, but with the additional possibility of infinitely interacting with the environment.

MPC-based: within this class of methods, we measure the performance of traditional optimization-based approaches using a Model Predictive Control (MPC) approach.

1) *MPC-Oracle*: this benchmark serves the purpose of quantifying the performance of an "oracle" controller. We provide this model with perfect foresight information of all future user requests and system dynamics within the planning horizon K. However, notice that this optimization model is NP-hard and thus does not scale well with increasing instance sizes.
2) *MPC-Forecast*: we relax the assumption of perfect foresight information in MPC-Oracle and substitute it with a noisy and unbiased estimate of demand. This method serves as a realistic control-based benchmark in situations where system dynamics are unknown. As for MPC-Oracle, this formulation suffers from the same lack of scalability.

B. Learning from Offline Data

We evaluate the rebalancing algorithms on taxi trip data from the cities of NYC (Brooklyn), Shenzhen (Downtown West), and San Francisco. We obtain the pre-processed data from [5]. We train each online baseline model for 10000 episodes with a time horizon $T=20$. As a planning horizon, we set $K=6$. For our offline RL experiments, we train each model on a fixed dataset of 10000 transitions.

Results in Table I show that by learning from a static dataset collected by a sub-optimal policy, the RL agent achieves performance on par with online methods, outperforming both heuristics and optimization-based benchmarks. Specifically, AMoD control policies learned through offline RL are only 7.2% (NYC), 5.5% (Shenzhen), and 10.7% (San Francisco) from Oracle performance, which assumes perfect information about future system dynamics. More importantly, the resulting policies are only 5.3% (NYC), 1.1% (Shenzhen), and 4% (San Francisco) from the performance of their online counterpart, which is allowed to freely interact with the urban network during training. Results in Table I also show how both RL-based approaches can achieve performance that is comparable or superior to that of MPC-Forecast, thus highlighting the benefits of using learning-based approaches to control stochastic, unknown systems, a characteristic that is particularly relevant in real-world settings where accurately modeling transportation demand is in itself a challenging process.

Crucially, the results highlight a drastic improvement in the sample efficiency of offline RL methods. If, on one hand, learning from online data requires approximately between 20000 and 200000 interactions with the transportation network, the offline agent is solely trained on a fixed dataset of 10000 samples, resulting in up to 20$\times$ improvement in sample efficiency. Most importantly, the samples used by the offline approach do not result from active interaction with the system but rather from historical data readily available to any service operator, thus challenging the current paradigm of simulation-based training and online interaction.

C. Impact of Dataset Quality

Within the offline RL literature, the fact that the performance of offline learning is highly dependent on the dataset characteristics is well-established [24], [25]. Hence, we conduct experiments using a variety of datasets by collecting training data from policies with different degrees of optimality [24], [21]. We store data generated by policies with approximately 75% (Medium M) and 90% (High H) and 100% (Expert E) of the online RL performance. Additionally, we collect a dataset via a deterministic, greedy heuristic, denoted by Greedy (G). The heuristic selects rebalancing trips to match the projected demand distribution averaged over the next K time steps.

The results reported in Table II confirm the strong dependency of offline learning on the dataset characteristics. Interestingly, a higher performance of the behavior policy does not indicate a better performance of the offline agent. Specifically, the policy trained on the Medium dataset outperforms the one trained on High in both NYC and Shenzhen scenarios. Moreover, despite being generated by a deterministic heuristic with a narrow policy, the G-dataset results in control policies that outperform the policy used to generate the data it was trained on. For example, the policy obtained via CQL in the Shenzhen scenario achieves improved rebalancing cost and revenue by 11% and 1%, respectively, compared to the greedy heuristic. The results demonstrate that all policies outperform the No Rebalancing and heuristic baselines, except for the agents trained on San Francisco-G and Shenzhen-G, suggesting that offline agents can produce policies that can be reliably deployed.

As an additional analysis, we report the relative reward of the behavior policy, together with the spread and 0.01-/0.99-interquartile

TABLE II

PERFORMANCE OF OFFLINE AGENTS TRAINED ON DATASETS WITH DIFFERENT CHARACTERISTICS. BEST-PERFORMING MODEL NOT TRAINED ON EXPERT DATA IS HIGHLIGHTED IN BOLD. REWARD INDICATES THE RELATIVE PERFORMANCE OF THE BEHAVIOR POLICY TO THE EXPERT POLICY.

	Performance			Dataset characteristics		
Dataset	Expert	Behavior Policy	CQL	Reward	Spread	IQR
NYC Brooklyn-M	56.1	42.6($\pm$2.4)	**53.1**($\pm$1.1)	76%	0.42	0.24
NYC Brooklyn-H	56.1	50.2($\pm$1.3)	52.0($\pm$1.3)	89%	0.16	0.13
NYC Brooklyn-G	56.1	49.6($\pm$0.9)	50.9($\pm$1.4)	88%	0.03	0.03
NYC Brooklyn-E	56.1	56.3($\pm$0.9)	56.4($\pm$0.8)	100%	0.12	0.11
Shenzhen West-M	62.3	45.8($\pm$3.6)	**61.6**($\pm$0.5)	74%	0.50	0.27
Shenzhen West-H	62.3	56.1($\pm$1.6)	59.7($\pm$0.9)	90%	0.25	0.15
Shenzhen West-G	62.3	55.9($\pm$0.7)	57.8($\pm$1.5)	89%	0.03	0.03
Shenzhen West-E	62.3	62.0($\pm$0.9)	61.9($\pm$0.7)	100%	0.13	0.11
San Francisco-M	14.8	11.6($\pm$1.0)	14.2($\pm$0.4)	78%	0.64	0.46
San Francisco-H	14.8	13.6($\pm$0.4)	**14.3**($\pm$0.4)	91%	0.47	0.27
San Francisco-G	14.8	13.1($\pm$0.5)	13.6($\pm$0.5)	88%	0.04	0.04
San Francisco-E	14.8	14.7($\pm$0.3)	14.8($\pm$0.3)	100%	0.28	0.24

range (IQR) of the actions in the dataset. This information is used to report the coverage (i.e., diversity) of the dataset. It should be noted that determining coverage in continuous state-action spaces is still an open challenge [25]; however, we believe the IQR to be a reasonable metric across the problem settings we investigate. This information provides insights into the inferior performance of the High- and Greedy-datasets for NYC and Shenzhen. Both datasets exhibit a narrow action distribution while still being far from expert performance. Hence, the agents are implicitly constrained to a limited subset of actions by the conservative loss on the Q-function, limiting the possibilities for improvement. Low coverage in the dataset can only be compensated by the high performance of the behavior policy, as in the case of the Expert dataset. Following the same line of reasoning, the superior performance of the Medium dataset can also largely be attributed to its diversity, where the difference in dataset coverage is a result of the method used for data collection. Specifically, as the training progresses, the Dirichlet distribution becomes increasingly narrow and the agent less exploratory. This finding is coherent with prior work, where CQL generally performs best on datasets with high coverage [25].

More broadly, we find that a fixed policy that hardly explores the state-action space is likely to be suboptimal for offline learning. In light of this, current operators should consider ways of gathering operational data while keeping in mind this quality-diversity trade-off. Moreover, since in this work, we assume that we do not have access to a simulation environment to validate the performance of our policy, we train our offline agents for a fixed number of steps. However, to strengthen the impact on real-world applications, employing an appropriate Off-Policy Evaluation (OPE) method [26] that allows us to validate the policy during training is essential. This approach will not only improve the final performance by enabling hyperparameter tuning and detection of overfitting but also provide an estimate for the policy performance and dataset quality before deployment.

D. *Transfer and Generalization*

To extensively assess the generalization capabilities of our proposed approach, we also study the extent to which policies learned through offline data are able to generalize to out-of-distribution (OOD) scenarios as well as to different cities. Specifically, the ability to transfer policies learned through offline RL to new environments without the need for additional data is particularly valuable for operators looking to expand their service to new cities. Table III presents the results of evaluating the zero-shot performance of a policy trained on offline data for the NYC Brooklyn dataset in comparison to (i) an agent trained online on NYC Brooklyn (i.e., SAC) and (ii) an expert policy, which is fully re-trained in the respective city (i.e., Expert). With zero-shot, we refer to the case in which policies are trained on data belonging to one city (e.g., New York) and later deployed to another system (e.g., Rome) *without further training*. Moreover, we assess the agents' ability to generalize in an OOD scenario within NYC-Brooklyn, where we simulate a surge in demand in an area that had low demand in the training data. The results show that the policy learned by CQL exhibits similar or better generalization performance compared to SAC trained on online data. Most importantly, despite being trained offline on data from a different city, agents learned via offline RL achieve an interesting degree of portability to new cities compared to the Expert policy, which has been re-trained from scratch in every environment. As observed in the previous sections, these experiments confirm how the transfer capabilities of offline policies are also highly dependent on the quality-diversity trade-off. This is evident in the case of the expert dataset (characterized by the smallest coverage but the highest reward), which leads to a policy that, although achieving the best performance in NYC, is outperformed by the CQL-M agent in other cities. In other words, learning from datasets that are more diverse, even if less performant, leads to agents that are better able to generalize across a variety of scenarios.

E. *Economically-feasible Reinforcement Learning*

In this section, we quantify the practical benefits of offline learning with subsequent online fine-tuning within AMoD systems compared to online RL methods. Specifically, although technically feasible, online data collection with a partially trained policy in AMoD systems is not realistic. Especially during the initial phases of training, the online agent requires extensive exploration of the environment, which will almost certainly result in erroneous and extremely high-cost rebalancing decisions. Within this context, offline RL offers a practically viable alternative by avoiding interactions with the environment until a sufficient level of performance is achieved.

The training curves depicted in Figure 2 and 3 clearly show

TABLE III

ZERO-SHOT PERFORMANCE OF SAC AND CQL TRAINED ON NYC-BROOKLYN.

City	Expert	No Reb.	ED	SAC	CQL-M	CQL-E
Rome	3.2(±0.2)	2.3(±0.2)	2.9(±0.2)	**2.9**(±0.2)	2.8(±0.2)	2.7(±0.2)
San Francisco	14.8(±0.3)	10.1(±0.4)	14.1(±0.4)	14.0(±0.5)	**14.2**(±0.4)	13.9(±0.5)
Shenzhen West	62.3(±0.7)	60.0(±0.7)	58.4(±0.9)	58.3(±3.1)	**60.0**(±1.0)	59.3(±1.2)
Washington DC	13.4(±0.2)	11.4(±0.3)	13.0(±0.2)	13.0(±0.8)	**13.2**(±0.3)	12.9(±0.5)
NYC-Brooklyn OOD	-	29.9(±1.1)	56.0(±0.2)	61.8(±1.2)	60.5(±1.4)	**62.1**(±1.1)

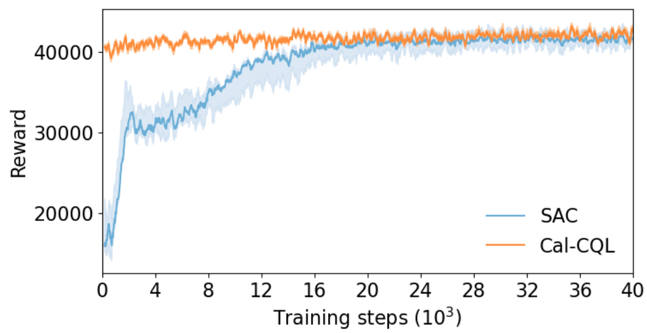

Fig. 2. Training reward obtained by an online agent compared to online fine-tuning of CQL-M in the NYC Brooklyn environment

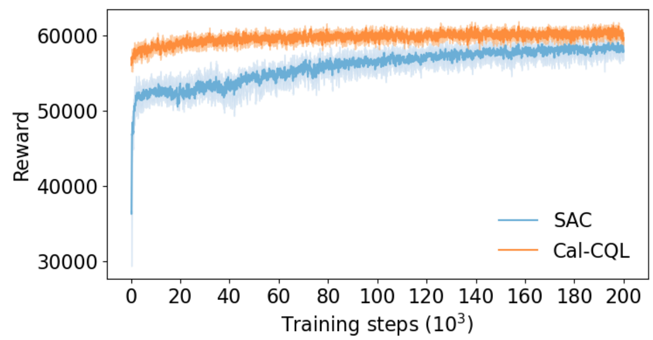

Fig. 3. Training reward obtained by an online agent compared online fine-tuning of CQL-M in the Shenzhen environment.

the extreme sub-optimality of the online learning paradigm. The offline learned agent successfully starts online fine-tuning in high-reward regions and avoids the low-reward regions that the SAC agent visits during initial exploration. Moreover, the inherent advantage of parametric policies - their tendency to become increasingly deterministic as training progresses - means that our offline pre-trained policy is inherently less stochastic, leading to safer online fine-tuning.

To further assess the potential savings offered by our approach, we compare the costs associated with training from scratch against the online fine-tuning of the offline trained agent in Table IV. We evaluate the difference in rebalancing costs and lost profit due to unmet demand using prices and costs estimated from real-world trip data. In NYC Brooklyn, the offline agent saves 20 780 interactions, over 2 600 000$ in rebalancing cost, and over 160 000$ of unmet demand, thus showing the significant amount of economical cost of training an agent within real-world systems. In the case of Shenzhen West, the pre-trained agent renders the entirety of online training from scratch redundant. Crucially, these experiments clearly quantify the benefits of offline learning and challenge the current approach of online learning by demonstrating its limitations.

TABLE IV

COST OF TRAINING AN AGENT FULLY ONLINE COMPARED TO ONLINE FINE-TUNING OF CQL-M.

City	Interactions	Reb. Cost	Unserv. demand
NYC Brooklyn	20 780	2 640 100	162 859
Shenzhen West	200 000	18 169 878	1 720 820
San Francisco	14 700	174 782	37 899

VII. CONCLUSION

Research on the central system-level control of AMoD systems focuses on developing RL approaches via online data. However, this is hardly practical within real-world urban systems, where the cost of making wrong decisions at a fleet level is extremely high. In this work, we propose offline RL as an appealing paradigm to approach the challenges in this space and enable centralized RL-based solutions to be applied even without simulated environments. Specifically, we introduce an algorithm that leverages the combination of offline RL and optimization to learn how to control AMoD systems solely via offline data, thus completely removing all interactions with both the real world and complex mobility simulators. Our approach shows strong performance in all real-world settings we evaluate, outperforming both optimization-based and heuristic approaches and matching the performance of online RL algorithms. Crucially, our method shows sample-efficient online fine-tuning capabilities, effectively enabling service operators to avoid expensive rebalancing decisions that make online training an impractical solution within real-world settings. In future work, we plan to investigate the combination of offline RL with ways to explicitly consider transfer and generalization in the design of neural architectures and training strategies, such as casting the problem under the lens of Offline Meta-RL. More generally, we believe this research opens several promising directions for the application of these concepts within economically-critical, real-world mobility systems.

REFERENCES

[1] U. D. E. S. Aff. (2021) 68% of the world population projected to live in urban areas by 2050. Available at https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.

[2] C. Mao, Y. Liu, and Z.-J. M. Shen, "Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach," *Transportation Research Part C: Emerging Technologies*, vol. 115, p. 102626, 2020.

[3] J. Feng, M. Gluzman, and J. G. Dai, "Scalable deep reinforcement learning for ride-hailing," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2060–2065, 2021.

[4] 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 *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2996–3003.

[5] D. Gammelli, K. Yang, J. Harrison, F. Rodrigues, F. Pereira, and M. Pavone, "Graph meta-reinforcement learning for transferable autonomous mobility-on-demand," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2913–2923.

[6] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," pp. 1–43, 2020.

[7] E. Candela, L. Parada, L. Marques, T.-A. Georgescu, Y. Demiris, and P. Angeloudis, "Transferring multi-agent reinforcement learning policies for autonomous driving using sim-to-real," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 8814–8820.

[8] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[9] M. F. 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*, 2018.

[10] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6019–6025.

[11] Z. T. Qin, H. Zhu, and J. Ye, "Reinforcement learning for ridesharing: An extended survey," *Transportation Research Part C: Emerging Technologies*, vol. 144, p. 103852, 2022.

[12] C. Fluri, C. Ruch, J. Zilly, J. Hakenberg, and E. Frazzoli, "Learning to operate a fleet of cars," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2292–2298.

[13] Y. Deng, H. Chen, S. Shao, J. Tang, J. Pi, and A. Gupta, "Multi-objective vehicle rebalancing for ridehailing system using a reinforcement learning approach," *Journal of Management Science and Engineering*, vol. 7, no. 2, pp. 346–364, 2022.

[14] Y. Jiao, X. Tang, Z. T. Qin, S. Li, F. Zhang, H. Zhu, and J. Ye, "Real-world ride-hailing vehicle repositioning using deep reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 130, p. 103289, 2021.

[15] X. Tang, F. Zhang, Z. Qin, Y. Wang, D. Shi, B. Song, Y. Tong, H. Zhu, and J. Ye, "Value function is all you need: A unified learning framework for ride hailing platforms," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3605–3615.

[16] B. Han, H. Lee, and S. Martin, "Real-time rideshare driver supply values using online reinforcement learning," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2968–2976.

[17] S. Sadeghi Eshkevari, X. Tang, Z. Qin, J. Mei, C. Zhang, Q. Meng, and J. Xu, "Reinforcement learning in the wild: Scalable rl dispatching algorithm deployed in ridehailing marketplace," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3838–3848.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870.

[19] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *ArXiv*, vol. abs/2203.01387, 2022.

[20] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1179–1191.

[21] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," 2020.

[22] M. Nakamoto, Y. Zhai, A. Singh, M. S. Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning," 2023.

[23] D. Gammelli, J. Harrison, K. Yang, M. Pavone, F. Rodrigues, and F. C. Pereira, "Graph reinforcement learning for network control via bi-level optimization," in *Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023.

[24] R.-J. Qin, X. Zhang, S. Gao, X.-H. Chen, Z. Li, W. Zhang, and Y. Yu, "NeoRL: A near real-world benchmark for offline reinforcement learning," in *Advances in Neural Information Processing Systems*, 2022.

[25] K. Schweighofer, M. Hofmarcher, M.-C. Dinu, P. Renz, A. Bitto-Nemling, V. P. Patil, and S. Hochreiter, "Understanding the effects of dataset characteristics on offline reinforcement learning," in *Deep RL Workshop NeurIPS 2021*, 2021.

[26] C. Voloshin, H. Le, N. Jiang, and Y. Yue, "Empirical study of off-policy policy evaluation for reinforcement learning," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, J. Vanschoren and S. Yeung, Eds., vol. 1. Curran, 2021. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/a5e00132373a7031000fd987a3c9f87b-Paper-round1.pdf