A Coverage Control-based Idle Vehicle Rebalancing Approach for Autonomous Mobility-on-Demand Systems

Pengbo Zhu, Isik Ilber Sirmatel, Giancarlo Ferrari-Trecate, Nikolas Geroliminis

Abstract—As an emerging mode of urban transportation, Autonomous Mobility-on-Demand (AMoD) systems show the potential in improving mobility in cities through timely and door-to-door services. However, the spatiotemporal imbalances between mobility demand and supply may lead to inefficiencies and a low quality of service. Vehicle rebalancing (i.e., dispatching idle vehicles to high-demand areas), is a potential solution for efficient AMoD fleet management. In this paper, we formulate the vehicle rebalancing problem as a coverage control problem for the deployment of a fleet of mobile agents for AMoD operation in urban areas. Performance is demonstrated via microscopic simulations representing a large urban road network of Shenzhen, China. Results reveal the potential of the proposed method in improving service rates and decreasing passenger waiting times.

Index Terms—Autonomous mobility-on-demand systems, coverage control, vehicle rebalancing, taxi fleet control

I. INTRODUCTION

THE continuous expansion of modern urban space and worldwide increasing population density in mega-cities lead to rapidly growing mobility demand. Addressing this problem exclusively via private vehicles is not sustainable owing to the increase in congestion and emissions. Moreover, public transportation alone cannot satisfy the need for timely, comfortable, and door-to-door mobility services. New trends in shared transportation are seen to have the potential to change the landscape and understanding of mobility. Mobility-on-Demand (MoD) systems (such as Uber, Lyft, and Didi) are promising solutions for providing passengers with efficient and fast service by deploying a group of coordinated vehicles serving ride requests within city areas. Using autonomous vehicles as MoD fleet (Autonomous Mobility-on-Demand, AMoD) is expected to improve upon these solutions [1].

The rapid adoption of MoD systems creates great challenges for city mobility operators and policymakers as there is limited information on the decision trends, safety implications, impact on vehicle ownership, and the effect of travel patterns on congestion. Furthermore, Transportation Network Companies

Pengbo Zhu and Nikolas Geroliminis are with the Urban Transport Systems Laboratory, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (email: `pengbo.zhu@epfl.ch`, `nikolas.geroliminis@epfl.ch`).

Isik Ilber Sirmatel is with the Control Section, Department of Electrical and Electronics Engineering, Faculty of Engineering, Trakya University, 22030 Edirne, Türkiye (email: `iilbersirmatel@trakya.edu.tr`).

Giancarlo Ferrari-Trecate is with the Dependable Control and Decision Group, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (email: `giancarlo.ferraritrecate@epfl.ch`).

(TNCs) are trying to attract more demand from other modes (e.g., taxis and buses) by offering shorter waiting times via using larger fleets, which can cause significant delays in the network due to a large number of circulating vehicles without passengers [2]. Thus, efficient MoD fleet management is an important direction for research, as it can enable TNCs to achieve the same service quality with smaller fleets, leading ultimately to less congested networks and thus shorter delays for all participants of urban mobility. Many studies have been investigated focusing on different aspects of MoD systems, for example, vehicle-sharing [3], routing [4], dynamic fare and pricing [5], and fleet sizing [6]. Moreover, in recent years, there has been an increasing interest in the spatial rebalancing problem of MoD fleets.

Asymmetry between origin and destination distributions of trips and non-uniform passenger demand for rides in different districts create discrepancies between actual and desirable spatial distributions of MoD fleets. For example, in the morning, people will typically travel from their houses to work, and in the evening, the trips will mostly be in the opposite direction. Obviously, more trips will begin or end around central business districts while relatively few will be in rural areas. Thus, proactively relocating idle vehicles to high-demand areas in real-time (i.e., vehicle rebalancing) can have significant improvements on MoD system performance [7], [8]. Nowadays, most TNCs use applications on mobile phones to receive requests from passengers and assign these orders to drivers [2]. Since a passenger will not wait indefinitely to be picked up, and drivers might refuse to answer a low-profit request, the imbalance of supply and demand can lead to long passenger waiting times and high cancellation rates, harming service quality. In MoD systems, there is a clear advantage in deploying vehicle rebalancing algorithms to achieve efficient operation by sending idle vehicles to districts with current or future high demand. Most literature and applications for rebalancing mainly focus on car-sharing and bike-sharing systems with much smaller fleet sizes [9], [10]. Bike-sharing systems are relatively easy to operate due to the size of bikes allowing a single mini-van to relocate a large number of bikes simultaneously [11], [12]. The management of car-sharing systems has been addressed at the strategic, tactical, and operational levels. For comprehensive reviews on these three planning levels, see [13]. The strategic and tactical decision levels relate primarily to station location [14], [15], fleet sizing [16], and staff sizing [17]. At the operational level, most of the emphasis has been put on devising methods to redistribute

vehicles in the system. However, the relocating of sharing bikes or cars mostly happens once per day at midnight, thus such methods are not suited for the real-time rebalancing of MoD systems.

Methods for MoD system management have been receiving increasing attention in the literature. In [18], by partitioning city districts into small areas (denoted as stations), a high-load case is presented to deploy a number of vehicles at each station for an AMoD system. Stations are modeled as vertices of a graph and a rebalancing policy is applied between vertices. A fluid discrete-time approximation model is introduced and an optimization approach is presented for steering the system to equilibrium [19], [20]. Model predictive control (MPC) approaches are employed to determine the optimal rebalancing policy in [21], [22], while [23] provides proof of stability within an MPC framework. [24] utilizes both historical and real-time data to build a prediction model. In [4], the rebalancing problem is described as an integer linear programming problem to match vehicles with passengers and balance the fleet. For shared MoD systems, a model-free reinforcement learning scheme is proposed to offset the imbalance in [25], while it has the risk of dispatching more vehicles than needed to an area. In most of the previous literature, a centralized controller is considered to operate the fleet. Moreover, as most works are based on region/station level, partitioning the city area into a limited number of virtual stations is required as a preprocessing step [26].

The application of advanced control methodologies is receiving attention for efficient management of traffic networks (see, e.g., [27]). For multi-agent systems, motion coordination problems have been investigated, which automatically distribute a fleet of mobile agents (such as autonomous vehicles or mobile robots) to carry out certain tasks in a bounded environment (e.g., rendezvous maneuver [28], task allocation, pattern formation [29]). Coverage control methods have been investigated to find the spatial configuration of agents that optimize a prescribed cost [30]. By assuming each mobile agent has a uniform circular sensing footprint, [31]–[33] focus on maximizing area coverage over the region of interest. The purpose of [34] is to minimize the sum of the demand-weighted distance between the agent and all points in its covered area. In [35], nonlinear dynamics of mobile agents were considered by using model predictive control schemes. Additionally, the coverage control problem has been expanded to consider constraints such as collision avoidance [36], energy consumption [37], and limited communication range [38]. The majority of previous works are based on dynamic Voronoi partition, whose Voronoi generators are the current positions of agents. In [34], Lloyd's algorithm [39] is used as a classic approach to drive agents to converge to a Centroidal Voronoi Configuration (CVC), with the center of mass of each Voronoi cell (i.e., centroid) coinciding with its generator [40]. The aforementioned works focus only on agents moving in a continuous Euclidean space. By considering a more practical setting, [41] and [42] use graph Voronoi partition [43] to represent a complex non-convex spatial environment.

There is great potential in applying coverage control to AMoD systems, as it can be operated in a distributed way with each agent computing its own control action, specifying better scalability properties compared to centralized schemes [44], [45]. Without pre-partitioning of the city area, the rebalancing position control can be made more precise via node-level (instead of region/station level) rebalancing. Furthermore, coverage control can run in real-time which is suitable for continuously allocating idle vehicles according to current idle fleet size and demand conditions.

Building upon our earlier conference work [46], this paper provides the following contributions:

1) To the best of our knowledge, it describes the first attempt at applying coverage control [34] to a real-time vehicle fleet rebalancing problem. The proposed method enables efficient AMoD system operation via dispatching idle vehicles towards high-demand areas, even the number of active vehicles involved in coverage is continuously changing due to passenger drop-offs and pick-ups. Furthermore, we validate the performance of the proposed method across various imbalance levels in trip origin and destination distribution, as well as different controller sampling times.

2) We extend the coverage control method to graphs (via [42]). This is important for the practical implementation of the proposed algorithm in urban road networks that invariably have a graph structure (while the original coverage control method involves a continuous space). The performance of these two methods is compared from different perspectives, e.g., computational complexity, implementation requirements, and so on.

3) We propose two kinds of upper-level controllers for choosing a subset of the idle vehicle fleet to hold position. This is required for reducing excessive rebalancing travel distance, e.g., back-and-forth motion, which might be caused by exactly following the coverage control actions in continuously changing traffic conditions. Using such an approach can mitigate the adverse effects of unnecessary traffic congestion and fuel consumption.

The remainder of the paper is organized as follows: Mathematical preliminaries of the coverage control method are introduced in Section II. Details of applying coverage control to vehicle fleet rebalancing are provided in Section III. In Section IV, simulation results, obtained using an MoD system simulator and synthetic scenarios with different imbalance levels, verify the proposed methods can answer more trip requests with less waiting time. Then, Section V considers the application of coverage control on the graph instead of on the continuous space. In addition, two controllers for determining active idle fleet size are introduced in Section VI, and some results are provided showcasing their operations. Finally, Section VII provides conclusions and potential directions for future work.

II. MATHEMATICAL PRELIMINARIES

In this section, we state the mathematical preliminaries and definitions of Voronoi partition and centroidal Voronoi configuration [40], and recall the control algorithm 'move-toward-the-centroid of each Voronoi cell' proposed in [34].

A. Voronoi Partition

Consider a bounded convex set $\Omega \subset \mathfrak{R}^2$. A number of n mobile agents are deployed, whose positions are $X = \{x_1, x_2, ..., x_n\}, x_i \in \Omega$ and their movements are constrained inside Ω. A naturally complete partition of Ω is the Voronoi partition, which gives a tessellation related to the responsibility area for an agent (Voronoi cell). Each cell consists of all points q in Ω that are closer to agent x_i than to any other agent. It is defined as

$$V_i(x_i) = \{q \in \Omega : \|x_i - q\| \leq \|x_j - q\|, \forall j \neq i\}. \quad (1)$$

where $\|\cdot\|$ is the Euclidean norm and x_i is called the generator/seed of the Voronoi cell $V_i(x_i)$ (as an example, $V_i(x_i)$ and $V_j(x_j)$ are shown as blue polygons in Fig. 1). A Voronoi partition $V = \{V_1, V_2, \ldots, V_n\}$ has the property that

$$\bigcup_{i=1}^{n} V_i = \Omega, \qquad int\{V_i\} \cap int\{V_j\} = \emptyset, \quad \forall i \neq j, \quad (2)$$
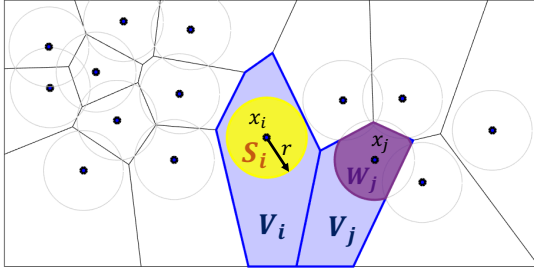
where $int\{V_i\}$ denotes the interior of V_i.

Fig. 1: Illustration of Voronoi partition.

B. Centroidal Voronoi Configuration

A continuous integrable function $\phi : \Omega \to \mathfrak{R}$ is termed a distribution density function and used for representing a measure of information or the probability of occurrence of an event at each point in Ω. For each Voronoi cell $V_i, i \in \{1, 2, \ldots, n\}$, the center of mass (i.e., centroid) is defined with respect to the density function ϕ as

$$C(V_i) = \frac{\int_{q \in V_i} q\phi(q)dq}{\int_{q \in V_i} \phi(q)dq}, \quad (3)$$

If the position of each agent $x_i, i \in 1, 2, \ldots, n$ is simultaneously located at its centroid, the corresponding pair (V, X) is called *centroidal Voronoi configuration* [40].

C. The Centroidal Coverage Control Law

The coverage objective function $H: \Omega \to \mathfrak{R}$ can be formulated as follows,

$$H(X, V) = \sum_{i=1}^{n} \int_{q \in V_i} f(\|x_i - q\|)\phi(q)dq, \quad (4)$$

where $f : [0, \infty) \to \mathfrak{R}$ is a nondecreasing differentiable cost which describes how the coverage performance degrades with the distance $\|x_i - q\|$ between an agent and a given point q. As an example, one can set $f(x) = x^2$, which is generally used in facility location problems [34].

We assume that agent $i \in \{1, 2, \ldots, n\}$ obeys the integrator dynamics,

$$\dot{x}_i = u_i, \quad (5)$$

where the agent velocity $u_i \in R^2, i \in \{1, 2, \ldots, n\}$ is the control input, which is computed as follows

$$u_i = -k_V(x_i - C(V_i)), \quad k_V > 0. \quad (6)$$

It is proved in [34] that a local optimum of coverage objective H will be achieved under Eq. (6), which makes the multi-agent system converge to the centroidal Voronoi configuration.

III. PROBLEM FORMULATION AND METHOD OVERVIEW

We consider an AMoD system consisting of a fleet of taxis, all of which are autonomous vehicles (AVs) having identical capabilities of sensing, communicating, computing and controlling their own motion. Moreover, we assume that the AVs have an unlimited communication range and thus they can detect any other adjacent taxis accurately. This is feasible technologically since the TNC platform can collect their real-time GPS coordinates and occupancy status. In this work, we assume that each AV can serve only one passenger at a time (or a unit of passengers with the same origin and destination, thus being functionally the same as a single passenger). Therefore the AV has the following three occupancy states:

- Idle (empty) vehicle: The vehicle is not carrying a passenger and is thus cruising/waiting for a passenger;
- Passenger-assigned: The vehicle is matched with a passenger and is moving to the passenger's origin position to pick up the passenger;
- Passenger-carrying: The vehicle has picked up its passenger and is moving to the passenger's destination position to drop off the passenger.

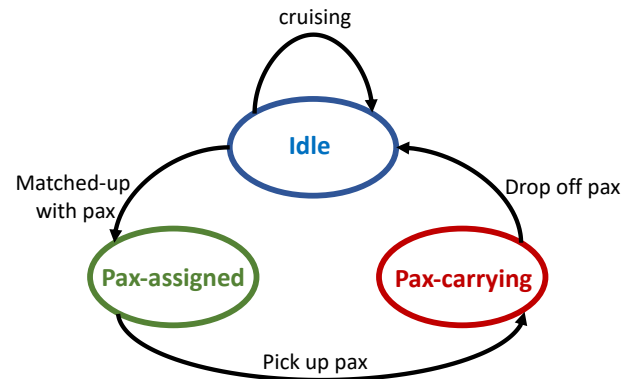The transitions between these occupancy states are illustrated in Fig. 2 .

Fig. 2: Finite-state machine schematic of an autonomous taxi. ('Passenger' is abbreviated as 'Pax' for brevity.)

The focus of this work is on developing a control algorithm for the efficient spatial rebalancing of the set of idle vehicles of an AMoD fleet. The algorithm is based on the coverage control algorithm proposed for robotic systems in [34], with the following intuitive reasoning: The pick-up task at a point $q \in V_i$ in the Voronoi cell V_i should be executed by the vehicle

closest to q, which is exactly the vehicle i by definition of Voronoi cell. Furthermore, the density function ϕ of coverage control is utilized here to describe the spatial intensity of mobility demand in the AMoD operation scenario.

A. Coverage control for Vehicle Rebalancing (CVR)

By implementing Eq. (3), the city area Ω is rasterized/discretized as a set of pixels, denoted as $\hat{\Omega}$. Let $|\hat{A}|$ stand for the number of pixel centers within the discretized area $\hat{A}$. Then the weighted centroid can be computed over the discretized Voronoi cell $\hat{V}_i$ (see Section 2.2 in [47]) as

$$C(\hat{V}_i) = \frac{\sum_{q \in \hat{V}_i} \hat{q} \phi(\hat{q})}{\sum_{q \in \hat{V}_i} \phi(\hat{q})}, \tag{7}$$

where $\hat{q}$ are the pixel centers. Computing Eq. (7) has a complexity of $\mathcal{O}(|\hat{V}_i|)$, while the worst case can be $\mathcal{O}(|\hat{\Omega}|)$. To limit the computational time, in contrast to [34], we additionally assume that each agent can cover a circular area with a limited radius r (shown as the gray circles in Fig. 1). For instance, the area covered by vehicle i (shown as the yellow disk in Fig. 1) is as follows

$$S_i(x_i, r) = \{q \in \Omega : \|q - x_i\| \le r\}. \tag{8}$$

At the same time, with the Voronoi partition, the actual area covered by vehicle i is $W_i = V_i \cap S_i$. For instance, for the example situation in Fig. 1, for vehicle i (shown at the point x_i), W_i is the same as S_i (yellow area), while for vehicle j, W_j is different to S_j (as shown by the purple area). Computing the centroid of $\hat{W}_i$ then has a complexity of $\mathcal{O}(|\hat{W}_i|)$ (which is $\mathcal{O}(|\hat{S}_i|)$ in the worst case). Similarly to the objective function (4) considered in [34], we formulate the coverage objective function H regarding r-limited Voronoi partition as follows,

$$H(X, W) = \sum_{i=1}^{n} \int_{q \in W_i} \|x_i - q\|^2 \phi(q) dq, \tag{9}$$

where $W = \{W_1, W_2, ..., W_n\}$.

For a region $W_i \subset \Re^2$, if we consider the probability density function ϕ as a mass density function, then the mass $M(W_i)$ is equal to $\int_{W_i} \phi(q) dq$. The centroid $C(W_i)$ is defined by (3) and the polar moment of inertia $J(W_i, x_i)$ are given by:

$$J(W_i, x_i) = \int_{q \in W_i} \|x_i - q\|^2 \phi(q) dq. \tag{10}$$

By still assuming the dynamics in (5), the control law for each vehicle i is defined as

$$u_i = -k_W (x_i - C(W_i)), \quad k_W > 0. \tag{11}$$

This control law guides every vehicle to move toward the centroid of W_i with a speed $\frac{k_W}{\|x_i - C(W_i)\|_2}$. In our discrete-time implementation, the coverage objective (9) is renewed at each time step according to the current empty vehicle coordinates, and the control action (11) is applied over a sampling period ΔT. As per Appendix A, any positive value of k_W guarantees a decrease of the objective function. Thus, even the value of speed may vary due to traffic congestion levels at different time steps, it does not violate the objective function's decreasing.

For the sake of brevity, the specification of the time step k is omitted in this section.

From the AMoD operation point of view, the control law operates the fleet by matching idle vehicle availability and trip demands via dispatching more empty vehicles to high-demand areas. Since it can be used for spatiotemporal deployment of idle vehicles of an AMoD fleet, we call the method Coverage control-based idle Vehicle Rebalancing (CVR) approach.

Remark 1: For each vehicle, the control algorithm has two steps: 1) Using local information (positions of its spatial neighbors and itself), compute the covered area W_i and weighted centroid $C(\hat{W}_i)$ (as an approximation of $C(W_i)$ according to Eq. (7)). 2) Move towards $C(\hat{W}_i)$. Therefore, the CVR method can be implemented in a distributed way (apart from the need for a central planner communicating demand density information to each vehicle). Moreover, the computation of the centroid scales linearly with $|\hat{W}_i|$ and therefore is suitable for real-time operations, enabling the rebalancing action for each agent to be updated in relatively short time intervals.
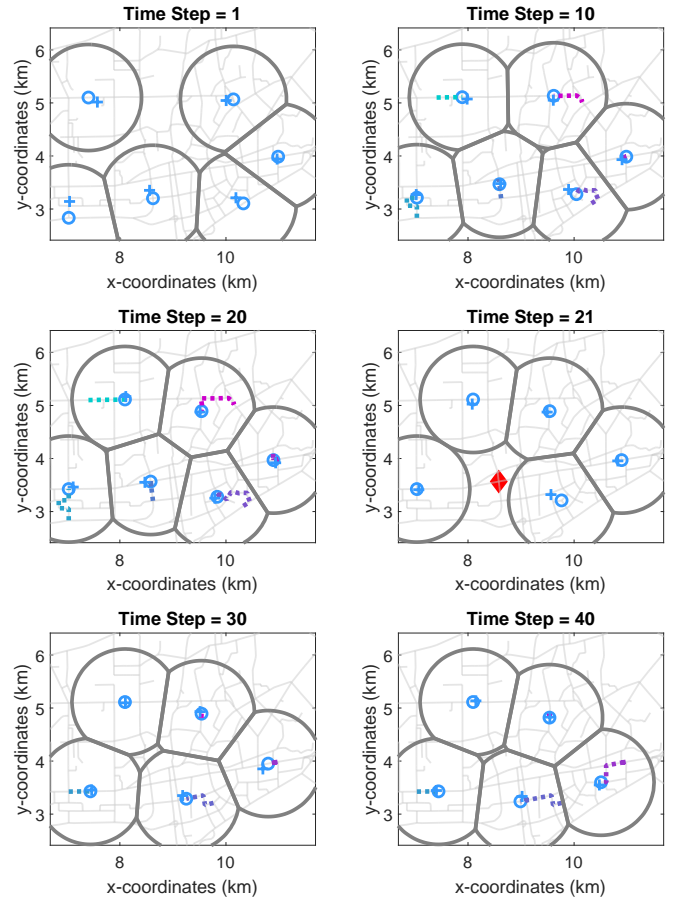

Fig. 3: A sequence of screenshots of vehicle rebalancing with CVR in action. 'o' marks the current positions of the vehicles, while '+' indicates their rebalancing goal positions (i.e., Voronoi centroids). The trajectories of these vehicles are shown by dotted lines.

Remark 2: Eq. (11) controls the vehicle position to follow gradient descent flow (see Appendix A for further details), which is not guaranteed to find the global optimum [34]. The

configuration is thus also affected by the initial position of agents. In contrast to previous works such as [34], due to the number of idle vehicles changing over time (as they can be assigned to, or drop off a passenger anytime), here the coverage control problem is not static. Instead, the proposed method involves solving at each time instant a different coverage control problem, the dimensions/structure (due to changing idle fleet size) and data (due to changing demand intensity) of which can change. For instance, we present a scenario where the system starts with 6 vehicles in Fig. 3, At Time step = 21, one vehicle (represented by a red diamond) gets assigned to a passenger, and is no longer considered in the Voronoi partition. Consequently, the remaining vehicles adjust and aim to converge to a new configuration based on a fleet size of 5. The final figure illustrates that the vehicle configuration dynamically adapts according to the current availability of vehicles. Notably, the intended goal of the proposed method is not to monotonically decrease the coverage objective and eventually converge to a locally optimal configuration. Instead, the proposed method attempts to continually steer the idle fleet towards configurations with a lower coverage value, thus enabling the AMoD fleet to operate with greater potential to have short response times, thereby increasing service quality.

B. Method Implementation

We use an undirected graph $G = (Q, E, w)$ to represent the city map, where E is the set of road links and Q is the set of vertices/nodes, i.e. intersections on city road, N is the number of nodes in G, and the edge weights w are the road length between two nodes. The proposed method is tested on an AMoD simulator replicating the urban road network of Luohu and Futian districts in Shenzhen, China. The network consists of $N = 1858$ intersections and 2013 road links.
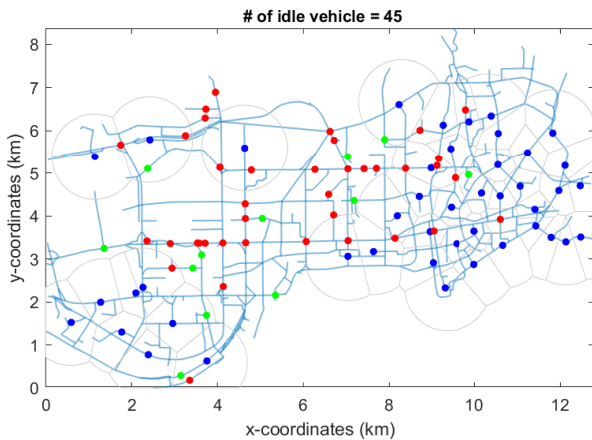

Fig. 4: A snapshot of the simulator. The idle/empty, passenger-assigned, and passenger-carrying AVs are dots in blue, green, and red, respectively. A demo video is available on YouTube: https://youtu.be/JlBs0CfuJ_c.

Fig. 4 shows a snapshot of the simulator. The blue dots stand for idle vehicles, and the gray contours denote the area covered by each vehicle (from the perspective of the coverage control objective function). In a practical setting, all vehicles should move along real urban roads and can only take turns or change their directions at the upcoming intersection. However, the calculated centroid $C(\hat{W}_i)$ may not be located on the roads that the vehicles can access. To circumvent this problem, instead of the computed exact centroid, we use the intersection/node closest to the centroid in the Euclidean metric as a simple approximation. Then, the implementation of the proposed CVR algorithm is provided in Algorithm 1.

Algorithm 1: Implementation of CVR.

At a time step $T(k)$,
for *each idle vehicle i, $i \in \{1, 2, \ldots, n_{idle}(k)\}$ (with current position $x_i(k)$, upcoming intersection is $Q^{next}(i,k) \in Q$, and its current rebalancing destination $Q^{dest}(i,k) \in Q$.* **do**
    Communicate: Transmit its position and obtain the position information of its adjacent neighbors.
    Compute: Compute its Voronoi partition and the centroid of its r-limited Voronoi cell $C(\hat{W}_i, k)$. Find the nearest intersection $Q^{dest}(i, k+1) \in Q$.
    Update: $Q^{dest}(i, k+1) \leftarrow C(\hat{W}_i, k)$
    Move: The vehicle moves towards $Q^{dest}(i, k+1)$ with speed given by Eq. (15).
end

IV. EXPERIMENT SETUP AND SIMULATION RESULTS

Supply and demand imbalances commonly exist in MoD systems, which leads to inefficient operation. These imbalances can be caused by the asymmetry between trip origin and destination distributions and heterogeneous demand levels in different regions. Thus, we will discuss the Origin-Destination (O-D) imbalance and the imbalance magnitude in Section IV-A and Section IV-B, respectively, and describe the passenger requests setting we use to test Algorithm 1 in Section IV-C. Then, the sensitivity analyses over different imbalance scenarios and fleet sizes are presented.

A. O-D imbalance

The historical data contains 199,819 taxi trips (collected during a 24-hour period) from the city of Shenzhen, China, consisting of their origin-destination pairs. In order to present the impact of the spatial discrepancy between trip origin and destination distributions, we consider both distributions to be fixed in the following discussions. Gaussian Mixture Models (GMMs) are used to estimate the spatial distributions of origins and destinations separately, as shown in Fig. 5. The origin distribution represents where the passengers would like to start their trips. At the same time, this continuous GMM of origin distribution is used as a demand density function ϕ of the coverage control algorithm, which satisfies the condition $\int_{q \in \Omega} \phi(q) dq = 1$.

Comparing Fig. 5a and Fig. 5b, it can be seen that these two distributions are different. Customer orders will gradually change the spatial distribution of supply (i.e., idle vehicle availability), since vehicles will move to respond to trip

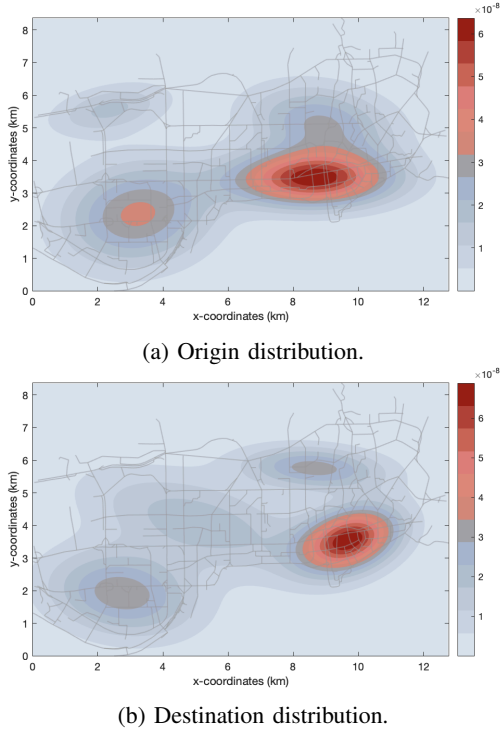(a) Origin distribution.

(b) Destination distribution.

Fig. 5: Contour map of Estimated Gaussian mixture models for the trip origin and destination distributions. More orders begin/end at the darker colored/red areas.

requests and tend to stay in the vicinity of drop-off (i.e., destination) points. Therefore, without any intervention, the supply distribution will shift towards the destination distribution (while ideally, it should match the origin distribution), resulting in severe spatiotemporal imbalances between supply and demand.

B. Imbalance parameter

In the considered AMoD system, all requests start and end at intersections, i.e., nodes on the graph. Using the historical data, we define the total number of pick-up requests at node i as $R^o(i)$, where $i = 1, 2, ..., N$. Then the probability that a pick-up happens at node i is as follows

$$p^o(i) = \frac{R^o(i)}{\sum_{i=1}^{N} R^o(i)}. \tag{12}$$

where $\sum_{i=1}^{N} p_o(i) = 1$. Similarly, we can get the probability of drop-off at node i as $p^d(i)$.

By defining the maximum of p^o as p_{max}^o, we can write

$$\tilde{p}^o(i) = p_{max}^o - p^o(i). \tag{13}$$

Then we normalize $\tilde{p}^o(i)$ over all nodes to have $\sum_{i=1}^{N} \tilde{p}^o(i) = 1$. We use $\tilde{p}^o$ as a complement of p^o, in order to generate synthetic destination distributions from origin distributions through a parameter $\gamma \in [0, 1]$ (representing O-D distribution balance) as follows

$$p_{\gamma}^d(i) = \gamma \cdot p^d(i) + (1 - \gamma) \cdot \tilde{p}^o(i), \tag{14}$$

where $\sum_{i=1}^{N} p_{\gamma}^d(i) = 1$. When $\gamma = 1$ the generated destination distribution p_{γ}^d is the same as the original one, i.e., p^d, while the smaller γ is, the more imbalance there is between the generated destination distribution and the origin distribution. When $\gamma = 0$, p_{γ}^d is the same as $\tilde{p}^o$, which has a shape complementary to p^o in that it generates the maximal O-D imbalance.

In addition, to test the effect of γ, we compute the Hellinger distance [48] between p_{γ}^d and p^o, which is a metric that takes values in $[0, 1]$ and it measures the degree of similarity between two probability distributions; when the distance is 0 the two distributions are identical and when it is 1 they are the furthest apart. The Hellinger distances between the generated destination distributions with different values of γ and the origin distribution are shown in Table I.

TABLE I: The Hellinger distances between p_{γ}^d and p^o for various values of γ

γ	0	0.25	0.5	0.75	1
Hellinger distance	0.2840	0.2391	0.1929	0.1421	0.0773

In the following sections, the origins of passenger demands are sampled from the origin distribution p^o from historical data, while the destination distribution follows p_{γ}^d generated by the above procedure.

C. Simulator Setup and Performance Metrics

In our simulations, we assume that the congestion is homogeneous across the urban area of interest, thus a macroscopic fundamental diagram (MFD, see [49]) is used to describe the relationship between the accumulation of vehicles m (i.e., number of all vehicles consisting of private ones and AVs) and the space-mean speed v (in m/s), as follows

$$v(m) = \begin{cases} 36e^{\left(\frac{-29m}{72000}\right)}, & m \leq 4320, \\ 6.31 - 2.33(m - 4320), & 4320 < m \leq 7200, \\ 0, & m > 7200. \end{cases} \tag{15}$$

The MFD is derived from traffic data, which can be collected through various kinds of sensors including loop detectors [50], GPS on vehicles [51], [52], drone-mounted cameras [53] and so on. Then the values of numerical parameters are estimated via curve fitting or system identification. Vehicle moving speeds are then emulated based on this MFD, given the vehicle accumulation. Note here, except extreme situations where network speed is zero (which is not included in the following simulation results), the magnitude of v is always greater than 0, which naturally satisfies the speed requirement in (11).

Similar to real MoD systems (such as Uber and Car2Go), the simulator match-up scheme obeys the 'first-come-first-served' policy. A request can be described by a tuple including the origin, the destination, and the time when the request is sent out. The matching process is described in Algorithm 2. The passenger waiting time consists of two parts: Time spent waiting to be matched with a vehicle, and the time spent waiting to be picked up. In reality, people have a limited

amount of tolerance to these, which we denote as t_{mtol} and t_{ptol} respectively. When passenger j sends out a request at time $t_0(j)$, the system will search for available idle vehicles, calculate the estimated pick-up time $\hat{t}_p(j)$ (according to current moving speed) for the spatially closest vehicle, and then match the passenger with this vehicle if the $\hat{t}_p(j)$ value is appropriate.

Algorithm 2: Matching process in AMoD system

Initialization: $N_{req} = 0, N_{order} = 0$
for *each passenger j (with the request sent out at time $t_0(j)$)* **do**
    $N_{req} \leftarrow N_{req} + 1$
    while $T(k) < t_0(j) + t_{mtol}(j)$ *min* **do**
        Compute: Find the spatially closest idle vehicle, compute pick-up time $\hat{t}_p(j)$
        if $\hat{t}_p(j) - t_0(j) \leq t_{ptol}(j)$
            Match the passenger with this vehicle,
            $N_{order} \leftarrow N_{order} + 1$,
            Break
        else
            $k \leftarrow k + 1$
    end
end
Return: N_{req}, N_{order}

If $\hat{t}_p(j) - t_0(j) > t_{ptol}(j)$, then the passenger keeps waiting for being matched for $t_{mtol}(j)$ min. At $(t_0(j) + t_{mtol}(j))$ min, if there are still no vehicles available that can respond to this passenger, she/he will cancel this request and leave the system. In this case, we assume that the passenger will choose to drive her/his private vehicle to travel, which will increase the accumulation m in the road network, leading to congestion.

If $\hat{t}_p(j) - t_0(j) \leq t_{ptol}(j)$, this vehicle will be assigned to the passenger (see the green dots in Fig. 4). Once the passenger and a vehicle are matched, we assume that this request cannot be canceled anymore and the 'request' becomes an 'order' for the vehicle. The vehicle then travels toward the origin position of the passenger to pick her/him up. After picking up the passenger, it will travel to the passenger's destination (see the red dots in Fig. 4). All traveling routes between any two nodes are pre-computed by the Floyd-Warshall algorithm [54] which yields the shortest path between any two nodes on a graph. The passenger-assigned and passenger-carrying vehicles have no contribution towards the coverage objective, while the proposed control method will continuously balance the idle vehicles to gravitate towards the high-demand areas, improving idle vehicle availability. Once a passenger arrives at her/his destination, the vehicle becomes idle again and joins the group of vehicles operated by the coverage control algorithm.

Considering that each vehicle might move in the network at various speeds at different times, we denote the time when passenger j is actually picked up as $t_p(j)$, and the total number of successfully completed orders as N_{order}. The average waiting-to-be-picked time can be computed as:

$$\bar{t}_w = \frac{\sum_{j=1}^{N_{order}}[t_p(j) - t_0(j)]}{N_{order}}. \qquad (16)$$

We define *completion rate* as the percentage of requests that are successfully completed as $N_{order}/N_{req} \times 100\%$. The $\bar{t}_w$ value has biases due to considering only the answered requests (i.e., orders) while ignoring the waiting-for-matching time of cancelled requests, so we introduce the average system time by giving a time penalty for the canceled requests as follows

$$\bar{t}_{sys} = \frac{\sum_{j=1}^{N_{order}}(t_p(j) - t_0(j)) + \sum_{j=1}^{N_{req}-N_{order}} \beta \cdot t_{ptol}(j)}{N_{req}}, \qquad (17)$$

where $(N_{req} - N_{order})$ is the number of canceled requests for the whole simulation, β is a weight parameter representing the time value penalty for cancelled orders (with $\beta > 1$). Note that the value of β does not affect the positive linear relationship between β and $\bar{t}_{sys}$, and it is chosen as $\beta = 1.5$ for the simulations. We choose a constant tolerance time for all passengers that $t_{mtol}(j) = 1$ min and $t_{ptol}(j) = 5$ min. Further simulation studies (omitted here for brevity) revealed that the system performance stops improving for r values larger than $r = 1000m$, thus we choose this value for the simulations.

D. Results and Analysis

In order to illustrate the influence of trip origin and destination imbalances, we generate synthetic scenarios with different values of imbalance parameter γ. Besides the spatial distribution, the arrival process of the passenger requests follows a Poisson distribution with piece-wise constant rates with a low-high-low demand profile where each period lasts for 1 hour. During the first and third hours (i.e., the low-demand periods), around 600 requests are issued per hour, while the arrival rate doubles during the second hour (i.e., the high-demand period). In total, around 2400 demand requests are introduced in a 3-hour simulation, and we assume the spatial distributions of requests are fixed. The baseline method, where vehicles stay at the destination of their last order until they are matched with their next passengers, is denoted as the *'Do-nothing' policy*. The proposed method CVR is compared with the Do-nothing policy using a set of 30 randomly generated simulation scenarios, and the results are provided in Fig. 6, showing the request completion rates, average waiting times and system times, with varying values of γ and fleet size, for the two methods. The solid lines present the mean value for the 30 runs, while percentiles are shown in shades for varying degrees ($25\%, 50\%, 75\%$, and 90%, respectively). The coverage control algorithm is operated every $\Delta T = 10$ s throughout this paper, unless otherwise specified.

According to Fig. 6, for all simulations, our method can improve system performance by yielding lower waiting and system times, and is able to serve more trips. These results indicate the potential of the proposed method in improving AMoD system performance by allocating more vehicles around high-demand areas and dynamically rebalancing their positions after dropping off passengers. When $\gamma = 1$, both methods do well due to the origin and destination distributions being similar and the situation requires little rebalancing effort. Even if there are no significant differences between the two methods regarding the completion rate (i.e., only an improvement of around 3%), our method shows substantially
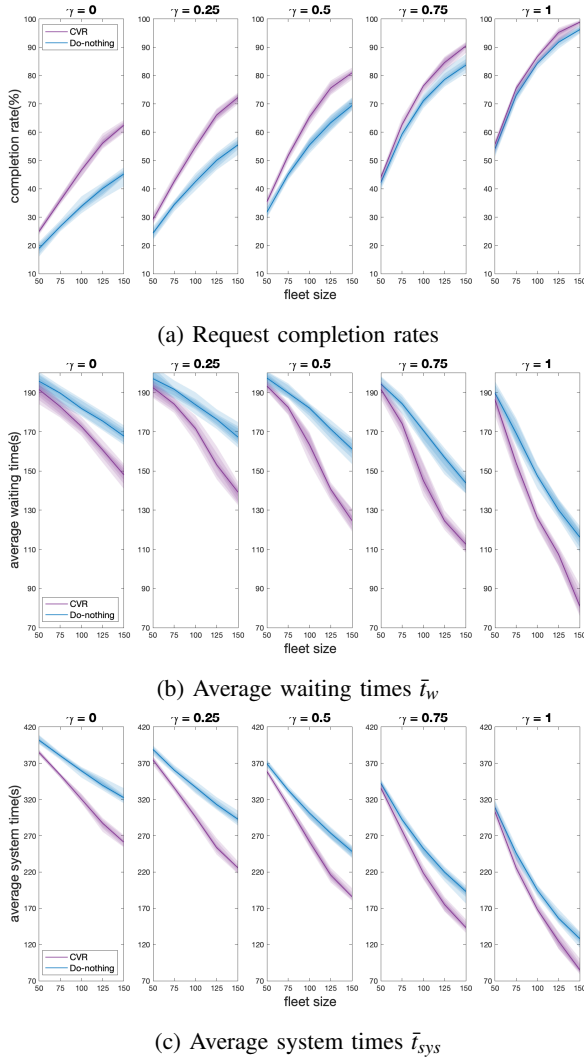
(a) Request completion rates

(b) Average waiting times $\bar{t}_w$

(c) Average system times $\bar{t}_{sys}$

Fig. 6: Comparison for various γ and fleet size values.

lower waiting and average system times, especially with large fleet sizes. With lower γ values, the performances of both methods decline as expected. With an increased imbalance, there will be more orders with origins and destinations further away from each other, causing the coverage control algorithm to require more time to steer the vehicles from their last destinations towards the high-demand areas. However, overall the simulations show that the proposed method yields better results than the baseline.

From the results, considering the varying fleet size values, it can be seen that the performance metrics improve for both methods with increased fleet size as expected. Furthermore, the proposed method can achieve similar or better performance with smaller fleets. For example, in Fig. 6b, for $\gamma = 0.5$ and a fleet size of 100, the proposed method can yield an average waiting time $\bar{t}_w$ of around 160 s, indicating an improvement of about 11% compared to the baseline of 180 s. In other words, to achieve a $\bar{t}_w$ around 160 s, the baseline requires a fleet size of about 150, which is 50 vehicles more than what the proposed method needs to achieve the same performance. Given that larger fleets are problematic due to increased costs for the TNCs and can also create additional congestion in

the urban network, the capability of the proposed method in achieving good performances with smaller fleets is a desirable feature for practical impact.

V. EXTENSION AND COMPARISONS

Although CVR on continuous space is easy to implement, the continuous space assumption suffers from several disadvantages: Firstly, the coverage control algorithm requires a continuous demand density function. Estimating a continuous density function from discrete historical trip data can introduce errors. Furthermore, areas accessible by a vehicle are confined to road networks in practical scenarios. Thus, the coverage region of one vehicle is non-circular and non-convex in shape (which can be thought of as the intersections and roads surrounding the vehicle), in contrast to the circular disk assumed in the original coverage control method (as described in Section II). Furthermore, the centroid calculated with the continuous Voronoi partition might be outside of any intersection on the city map, thus the closest intersection is used as an approximate position of the exact centroid. These steps inevitably introduce approximation inaccuracies. To overcome these problems, as the city map can be modeled as a graph, the original coverage control algorithm can be extended to coverage on graph algorithm (following [42]) using the graph Voronoi partition method of [55].

A. Problem Reformulation

The urban road network is modeled as an undirected graph $G = (Q, E, w)$. For the vehicle i, whose current position is x_i, the Voronoi tessellation of graph is given by the cells [55]

$$V_i^G(x_i) = \{q \in Q : d(x_i, q) \le d(x_j, q)), \forall i \neq j\}, \quad (18)$$

where $d(A, B)$ stands for the shortest distance between nodes A and B in graph G.

Considering concerns related to computational complexity, similar to the continuous case where an r-limited Voronoi cell is used to depict the covered area of each agent, the graph coverage control scheme is implemented by bounding the graph Voronoi cell within r^G. Taxi traveling on dense urban roads can be approximated by motion on a grid map, to relate the coverage on the graph to continuous space, here we choose $r^G = \sqrt{2}r$, which measures the coverage radius r in Eq. (8) on the grid map by Manhattan Distance, i.e., the shortest path that a taxicab would take between city intersections), which is widely used in path planning [56]. For the covered nodes of each idle vehicle i, we only consider the nodes within its graph Voronoi cell which are not further than r^G limits from its position x_i. Therefore, we obtain the 'covered area' of each idle vehicle on the graph as

$$S_i^G(x_i, r^G) = \{q \in Q : d(q, x_i) \le r^G\}. \quad (19)$$

The r^G-limited graph Voronoi cell of vehicle i is then $W_i^G = V_i^G \cap S_i^G$. A discrete version of the objective function Eq. (9) can be formulated as:

$$H(X, W^G) = \sum_{i=1}^n J_i^G(x_i, W_i^G) = \sum_{i=1}^n \sum_{q \in W_i^G} d(x_i, q)^2 \phi^G(q), \quad (20)$$

where the demand density ϕ^G is a probability mass function over the nodes in G, which is the same p^o introduced in Section IV-A. It satisfies

$$\sum_{q \in Q} \phi^G(q) = 1. \tag{21}$$

According to [57], the centroid of r^G-limited graph Voronoi can be computed by an integer optimization problem as

$$C(W_i^G) = \arg \min_{q \in W_i^G} J_i^G(x_i, W_i^G). \tag{22}$$

If the feasible solution of Eq. (22) is not unique, we randomly select one element from the solution set.

Fig. 7 is a snapshot of the r^G-limited graph Voronoi partition. For instance, for the vehicle in x_i, the covered nodes are shown as blue squares, and the graph centroid is denoted as $C(W_i^G)$. This idle vehicle i will travel towards $C(W_i^G)$ following the shortest path calculated by the Floyd-Warshall Algorithm. We name this control algorithm as CVR-graph.
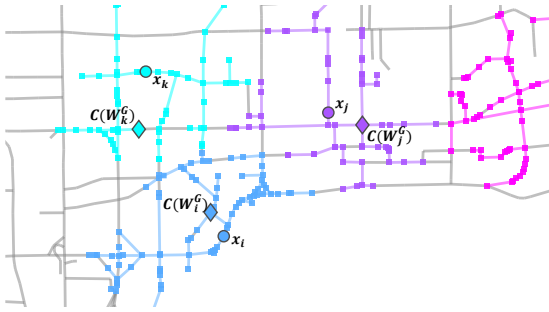
Fig. 7: A snapshot of partition on graph of urban road. For each vehicle (round markers), the square markers in the same color around it stand for the graph nodes which are closer to it than to any other within a distance of r^G. The diamond marker illustrates the position of its centroid where this idle vehicle should move to.

Remark 3: In contrast to the CVR on a continuous space, there is no need to estimate the demand density with CVR-graph. The partition on the graph can cover non-convex regions as required for practical coverage. The centroid can also be directly obtained by Eq. (22) without approximation. The disadvantage of the CVR-graph method is the increased computational effort. On the graph of urban roads, the edge weights are not the same as they reflect the length of road segments. For this non-tree graph, computing the centroid is the most time-consuming part, and the core of it is computing the distances between one node q and all the other nodes in W_i^G (one-to-all distances) which requires $\mathcal{O}(|W_i^G| log |W_i^G|)$ by Dijkstra's algorithm. Please note, by slight abuse of notation, in this section, $|A^G|$ stands for the number of all nodes on graph 'covered' by A^G.

B. Results and Comparative Analysis

We introduce a comparison of our approaches with a fleet rebalancing strategy detailed in [3]. This strategy focuses on relocating idle vehicles to intersections where there are pending or unassigned requests. The strategy operates through linear programming (LP) that aims at minimizing the total travel time between pairs of vehicles and requests, subject to the condition that either all pending requests or all idle vehicles are allocated. While our proposed methods, CVR and CVR-graph, control the fleet in a distributed manner, the LP strategy from [3] operates in a centralized way and requires real-time access to information regarding all unassigned requests.

An experiment is carried out under origin-destination trip demand imbalances (with $\gamma = 0.5$) and a fleet size of 150. Around 2400 requests are introduced in a 3-hour simulation to compare the performances of CVR on the continuous map and on the graph. The control sampling time of all policies is set as $\Delta T = 10$ s. The performance metrics are listed in Table II.

TABLE II: Performance metrics of different policies

	completion rate (%)	waiting time (s)	system time (s)
CVR-graph	82.7	125.7	181.9
CVR	82.7	127.4	183.1
LP [3]	80.8	150.8	208.1
Do-nothing	72.3	157.6	238.5

Compared with both LP and Do-nothing policies, CVR-graph yields better results: Answering more requests, at the same time, gives a clear reduction in average waiting and system time. The results demonstrate CVR-graph can help to improve the efficiency of the AMoD system.

Furthermore, we compare CVR-graph to CVR proposed in Section III. In Fig. 8, the trajectories of vehicles under CVR are depicted. The vehicles move towards high-demand regions. Notably, Vehicles 9 and 10 show back-and-forth movements, potentially due to centroid approximation errors. For instance, while the actual centroids (red '+') for Vehicle 9 at time steps 8 and 16 are proximal, both of them lie outside intersections on the graph. Then the closest intersections (orange crosses) are used as approximate centroids, i.e., their rebalancing destinations. It results in a change of destination, causing Vehicle 9 to move westward first (as seen at Time Step 8), reverse its course (Time Step 16), and then head west again (Time Step 24). A similar pattern is observed with Vehicle 10. In Fig. 9, the trajectories of vehicles operating under CVR-graph are presented. In contrast to CVR, there is no observed oscillatory movement, thanks to the exact centroid computation.

According to the performance metrics as listed in Table II, the CVR-graph is seen to yield similar performance to CVR, with a slightly shorter waiting time. This might be attributed to the studied city map, where the distribution of graph nodes is dense and roughly uniform. Consequently, approximating this road network with a continuous space turns out to be a reasonable approximation, as can be evidenced by Fig. 8 showing that the exact centroids (red) and the approximate centroids (orange) are close. Although CVR and CVR-graph yield similar performances in studied dense road networks, discrepancies between these two methods could become more significant when vehicles are operated in dedicated lanes or utilize only a specific subset of the road network.
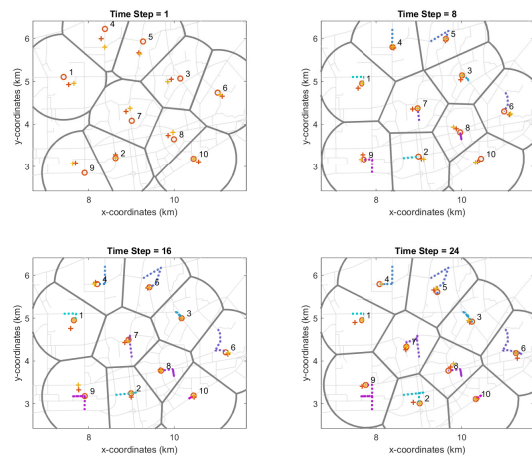
Fig. 8: A sequence of screenshots with CVR in action. 'o' are the current positions of vehicles, red '+' show their exact Voronoi centroids, and orange '+' are approximate centroids (true rebalancing destination/goal positions). The dotted lines represent their trajectories.
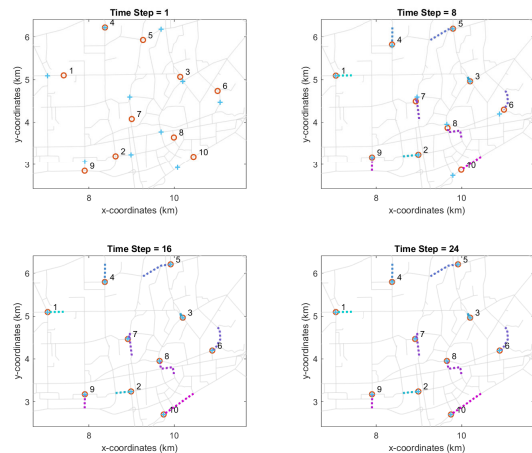

Fig. 9: A sequence of screenshots with CVR-graph in action. 'o' are the current positions of vehicles, blue '+' show their centroids of graph Voronoi cell. The dotted lines represent their trajectories.

To sum up, CVR involves approximations that can lead to inaccuracies and potentially unnecessary oscillatory movements. In contrast, CVR-graph, though computationally relatively more intensive, provides more accurate and practical guidance for vehicle deployment.

C. Performance across different controller sampling time

We test the performance of our methods, CVR and CVR-graph, and the LP policy for a set of control input update period (i.e., ΔT, controller sampling time) values ranging between 10 and 300 seconds. The results are shown in Fig. 10.

It can be seen that performance deteriorates for all methods as the controller sampling time increases, because the system becomes less responsive and fewer vehicles are rebalanced to high-demand regions as expected. Our methods consistently outperform the LP policy, addressing more requests and re-
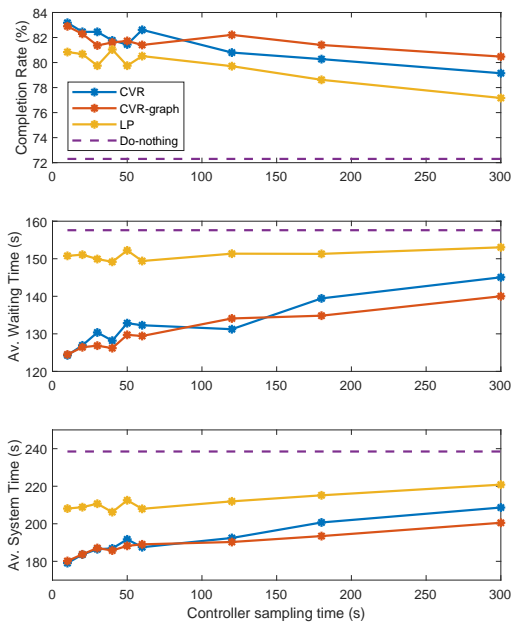

Fig. 10: Performances of CVR, CVR-graph, LP policy and Do-nothing policy as functions of controller sampling time.

ducing waiting times across all tested scenarios. LP policy bases its control actions on the unfulfilled orders from the previous control sampling period, thus a longer sampling time results in less timely actions. For short controller sampling times ranging from 10s to 60s, CVR and CVR-graph show comparable completion rates, average waiting, and system time. However, when the sampling time is extended, CVR-graph demonstrates superior performance over CVR. While the performance metrics of answer rate and average waiting time for CVR and CVR-graph deteriorate as the sampling time increases, they remain consistently better than those of the LP policy and Do-nothing policy.

VI. ACTIVE IDLE FLEET SIZE ADAPTATION

Outside of rush hours, there might be many idle vehicles circulating without serving any passengers. Furthermore, the proposed coverage control algorithm might force idle vehicles to execute many short back-and-forth movements due to small changes in idle fleet size inside short time intervals. The resulting empty kilometers traveled cause fuel waste, air pollution, and congestion. We therefore investigate two different methods for selecting a subset of idle vehicles not to be relocated. By contrast, the rest of the idle vehicles (which are actively repositioning themselves according to the coverage control law) are termed 'active idle' vehicles. The goal of these methods is to reduce empty travel distances caused by mostly unnecessary small rebalancing actions.

There exist two questions in the idle fleet size selection problem: 1) How many vehicles should be relocated for different demand levels, and 2) which ones should be chosen for relocation?

Firstly, for each idle vehicle i (with current position x_i), the polar moment of inertia of its covered area is given by

Eq. (10). Similarly, we can define the polar moment of inertia of its Voronoi cell as

$$J(V_i, x_i) = \int_{q \in V_i} \|x_i - q\|^2 \phi(q) dq. \tag{23}$$

To guarantee service quality, we prefer to select vehicles that are located in high-demand areas and ask them to hold their positions. In a map with non-uniform density, with the proposed coverage control scheme, more idle vehicles will gather around high-demand regions (see Fig. 4). This implies that a vehicle located at a high-demand area usually has a higher value of $J(W_i, x_i)/J(V_i, x_i)$, since owing to the nature of Voronoi partition, W_i will cover more area of the whole V_i. Note that both 'not-relocated idle' and 'active idle' vehicles should be considered for Voronoi partitioning (the contours of their covered areas are shown in gray in Fig. 11).
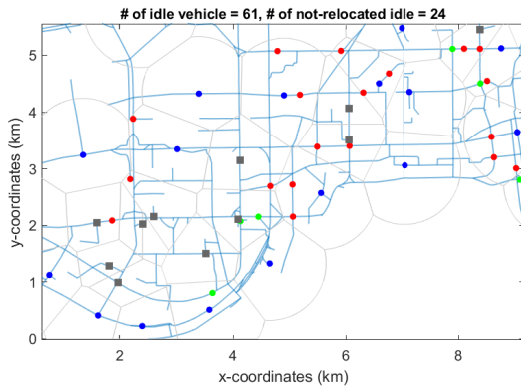

Fig. 11: A partial screenshot of the simulator. The 'active idle' vehicles are shown as blue dots, while the 'not-relocated idle' vehicles are shown as gray squares.

A. Active idle fleet size adaptation controllers

We introduce two different approaches to determine how many vehicles should not be relocated: 1) CVR-α simply chooses α percent of idle vehicles, 2) CVR-PI uses a proportional-integral (PI) controller which collects and uses the passenger average waiting time in a time window of the immediate past.

1) CVR-α:

At each time step k, the number of idle vehicles is $n_{idle}(k)$. For CVR-α, $\lfloor n_{idle}(k) \times \alpha \rfloor$ idle vehicles with the largest $J(W_i, x_i)/J(V_i, x_i)$ value will be selected and stopped at their current positions, where $\lfloor x \rfloor$ is the greatest integer less or equal to x. Here we test $\alpha = 0\%, 20\%, 40\%, 60\%, 80\%, 100\%$. For $\alpha = 0\%$, one has the same setting as CVR, which never lets any idle vehicle become 'not-relocated'. The case $\alpha = 100\%$ corresponds to the 'Do-nothing policy', which means the empty vehicle stops at its last destination where it drops off the last passenger until it is matched with the next passenger.

2) CVR-PI:

The goal is to design an adaptive method to determine how many vehicles should hold their position. A PI-controller using a constant reference signal is introduced here to that end.

The fleet size adaptation controller is operated at a constant frequency, which is chosen as once every $\Delta T' = 5$ min. At each time step k', we collect and calculate the average waiting time $\bar{t}_w(k')$ and the average number of empty vehicles $\bar{n}_{idle}(k')$ in the last $\Delta T'$ minutes. The output signal can be obtained as $y(k') = \sqrt{\bar{t}_w(k') \cdot (n_{AV} - \bar{n}_{idle}(k'))}$, where n_{AV} is the number of all vehicles in the AMoD system.

The error signal is defined as the difference between a constant reference and the output as

$$err(k') = y_{ref} - y(k'). \tag{24}$$

Then, the discrete PI controller is given by

$$\Delta u_{not}(k') = K_p \cdot err(k') + K_I \sum_{i=1}^{k} err(k'), \tag{25}$$

and $u_{not}(k+1)$ can be calculated as

$$u_{not}(k'+1) = u_{not}(k') + \Delta u_{not}(k'). \tag{26}$$

In the next time step, $\lfloor u_{not}(k'+1) \rfloor$ idle vehicles with the largest $J(W_i, x_i)/J(V_i, x_i)$ values are selected as 'not-relocated idle vehicles', and the rest as 'active'. More information on tuning the parameters of CVR-PI can be found in Appendix B.

B. Simulation Results

Here we define a performance metric for evaluating the proposed schemes: the *accumulated rebalancing distance*, which is the total traveling distance caused by relocating active idle AVs. Obviously, the rebalancing distance of Do-nothing policy is always 0 due to all empty vehicles stopping at their last destination.

Next, we present a simulation experiment over 3 hours under origin-destination trip demand imbalances $\gamma = 0.5$ with a fleet size of 150 vehicles. We compare the performances of our proposed methods with the Do-nothing policy. The performance metrics are provided in Table III. For CVR-PI, we have set $K_p = 0.2, K_I = 0.4$ and $y_{ref} = 60$.

TABLE III: Performance of fleet size adaptation methods.

|	completion rate (%)	$\bar{t}_w$ (s)	$\bar{t}_{sys}$ (s)	accu. rebalancing distance (km)
CVR($\alpha = 0\%$)	82.6	127.0	183.2	4804.8
CVR-PI	82.4	139.4	194.2	2314.7
CVR-$\alpha = 20\%$	82.2	132.5	188.9	3890.1
CVR-$\alpha = 40\%$	80.9	135.5	195.5	2960.0
CVR-$\alpha = 60\%$	80.1	137.6	199.7	2031.6
CVR-$\alpha = 80\%$	78.6	146.5	211.5	1047.5
Do-nothing	73.1	155.1	234.5	0

It can be seen from Table III that CVR yields the highest completion rate and the shortest waiting time. However, since all vehicles are active and relocated all the time, the accumulated rebalancing distance is the highest. For CVR-α, for higher values of α, the waiting time becomes larger and the completion rate tends to decrease slightly. Generally speaking, the fleet size should be chosen properly to strike the desired trade-off between waiting time and extra traveling caused by rebalancing. It is interesting to note, however, that the CVR-PI can finish almost as many orders as CVR with the rebalancing distances reduced by 51.8%. Furthermore, compared to the
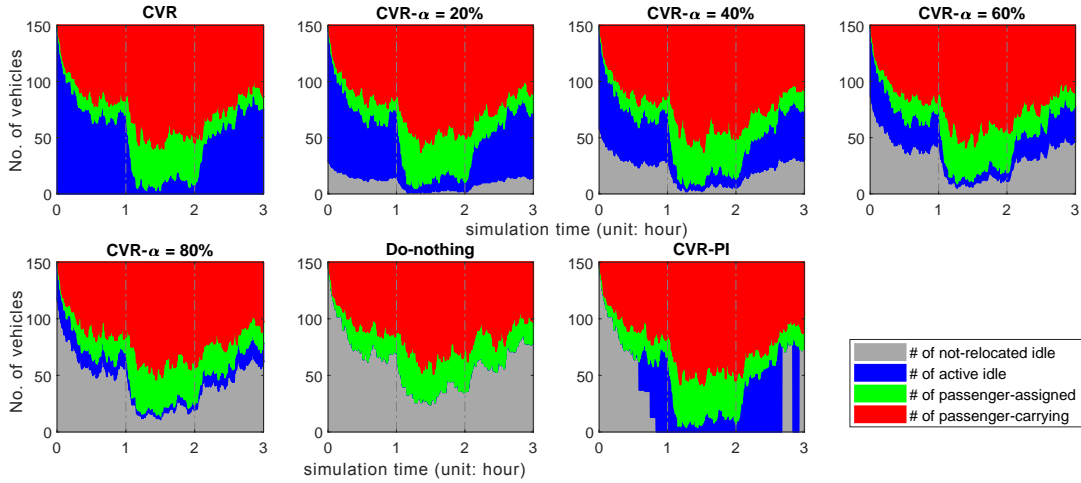
(a) No. of vehicles in different states
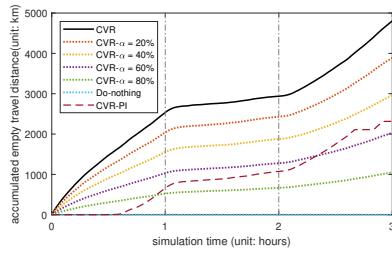
(b) Accumulated rebalancing distances
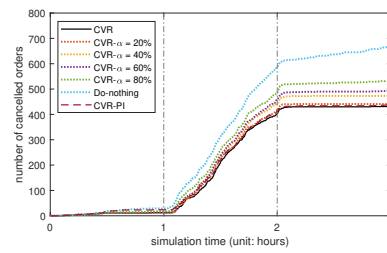
(c) Accumulated canceled orders

Fig. 12: Comparison of CVR, CVR-α, and CVR-PI.

Do-nothing policy, it can still yield significant improvements in waiting time and system time. These results indicate that the CVR-PI method is capable of achieving a good balance between system performance and rebalancing costs.

To demonstrate the simulation process more precisely, the results in Fig. 12a depict the time evolution of the number of vehicles in various operating states (i.e., not-relocated idle, active idle, passenger-assigned, and passenger-carrying). In Fig. 12b and Fig. 12c, the solid lines present the CVR method, while results of the CVR-α are shown by dotted lines and CVR-PI in brown dashed lines.

During the first and second hours, it can be seen from the evolution of vehicle states that the coverage control approach helps the system as the sum of blue and gray areas (i.e., the number of active idle vehicles + the number of not-relocated idle vehicles) of all algorithms are smaller than Do-nothing in Fig. 12a. It indicates that the proposed methods are able to operate the fleet more efficiently, as a larger amount of vehicles is actively serving passengers most of the time. On the other hand, the Do-nothing policy operates with a larger amount of idle vehicles, which manifests itself as a greater amount of canceled orders, as can be seen from Fig. 12c.

Fig. 12a shows that, at the beginning and the end of low-demand periods, CVR-PI method keeps more idle vehicles static; while during the high-demand hour, all vehicles are actively doing coverage control which guarantees completion rates as good as CVR, however with reduced rebalancing distances.

In Fig. 12c, during rush hours, a steeper upward tilt to the curve for Do-nothing policy can be observed. After the high-demand period, the number of canceled orders for the Do-nothing policy still keeps increasing, while except for CVR-$\alpha = 80\%$ which produces a few more canceled orders, the other methods can manage to respond to new requests almost without any more cancellations.

VII. CONCLUSIONS

In this paper, we proposed the application of coverage control to rebalance vehicle fleets for autonomous Mobility-on-Demand systems. Being a model-free approach, it provides a relatively simple way to give online node-level guidance for idle vehicle fleets. For compensating spatiotemporal imbalances in demand and supply, our proposed method can dynamically rebalance the spatial distribution of idle vehicles to serve more trips with reduced waiting times. To treat the non-circular areas covered by vehicles in road networks due to road geometry, we extended the original coverage control algorithm to the graph Voronoi partition. Considering the empty traveling distance due to the repositioning process, we proposed different approaches to automatically manage the active idle fleet size. Our coverage control algorithms are tested on a discrete city map using real road network geometry and trip data from the city of Shenzhen. The performances of the proposed approaches are compared with a linear programming-based rebalancing approach and a 'Do-nothing' policy, and the results show clear advantages.

Simulation results depend on a pre-computed static demand density function which is estimated from historical data. However, the demand patterns are highly influenced by weather and other external effects in reality, also demand data can be skewed by measurement errors. Future research will consider using time-varying demands based on real-time trip request data for fleet rebalancing. For reference on integrating estimating density function with coverage control, see works [58], [59]. Additionally, it is more practical to consider a mixed system consisting of both autonomous vehicles and human drivers, with the latter potentially having inaccuracies in following the rebalancing control actions. Then besides system-wide metrics, investigating the profit of individual drivers as discussed in [60] is one interesting topic. Another limitation of the current control schemes is that the fleets lack coordination with other regions in the network. One way to solve this problem is to develop a hierarchical control architecture. It can benefit from efficient coordination between the actions of upper-level controllers which operate the aggregated traffic components (for example, how many idle vehicles should travel from one sub-region to another) and those of the lower-level controllers based on the proposed CVR approaches.

APPENDIX A
PROOF OF COVERAGE CONTROL LAW

This section provides the proof that the coverage control law (11) can decrease the objective function $H(X,W)$ and asymptotically steer the agents to the centroids of their r-limited Voronoi cells. The proof is largely inspired by [34].

According to the parallel axis theorem, similarly to [34], we can rewrite the polar moment of inertia as

$$J(W_i, x_i) = J(W_i, C(W_i)) + M(W_i)\|x_i - C(W_i)\|^2, \quad (27)$$

where $J(W_i, C(W_i))$ is the polar moment of inertia with respect to the centroid $C(W_i)$.

From Eq. (9) and Eq. (27), the objective function and its partial derivative with respect to x_i can be expressed as:

$$\begin{aligned} H(X,W) &= \sum_{i=1}^{n} J(W_i, x_i) \\ &= \sum_{i=1}^{n} J(W_i, C(W_i)) + \sum_{i=1}^{n} M(W_i)\|x_i - C(W_i)\|^2, \\ \frac{\partial H(X,W)}{\partial x_i} &= 2M(W_i)(x_i - C(W_i)). \end{aligned}$$
$$(28)$$

From the first-order condition $\partial H(X,W)/\partial x_i = 0$, one has that the optimal configuration involves positioning all agents at the centroids of their respective r-limited Voronoi cells, i.e. $x_i = C(W_i)$.

The control algorithm should minimize the coverage cost H by properly allocating n agents in space. To achieve this

goal, the control law (11) makes the position of each agent x_i follow a gradient descent flow.

For the system given by (5) and (11), the objective function H is guaranteed to decrease for $\forall k_W > 0$ because

$$\begin{aligned} \frac{\mathrm{d}H}{\mathrm{d}t} &= \sum_{i=1}^{n} \frac{\partial H(X,W)}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}t} \\ &= -2k_W \sum_{i=1}^{n} M(W_i)\|x_i - C(W_i)\|^2 < 0. \end{aligned} \quad (29)$$

APPENDIX B
PI CONTROLLER REFERENCE TUNING

In this section, we provide the parameter tuning process of the PI controller for the proposed CVR-PI method. With $y(k')$ reflecting the service level in the last 5 minutes, when $y(k')$ is small the fleet of active vehicles can be deemed to fulfill the current requests. In this case relocation of idle vehicles is not highly beneficial, thus when $y(k') \leq 90$, all idle vehicles will hold their current positions. Otherwise, i.e., when $y(k') > 90$, we use a PI controller to track a constant reference y_{ref} to determine how many vehicles should not relocate. Next, we describe the result of a sensitivity analysis that we performed for tuning y_{ref}.

Under the same experiment settings of Section IV, we use different values of y_{ref} (from 30 to 120 with an increment of 10) and collect the performance metrics. The tuning criterion (denoted as θ) is a normalized combination of two performance criteria, and is computed by summing up the normalized average system time (scaled to $[0,1]$) and normalized rebalancing trips (scaled to $[0,1]$). The results are shown in Fig. 13.
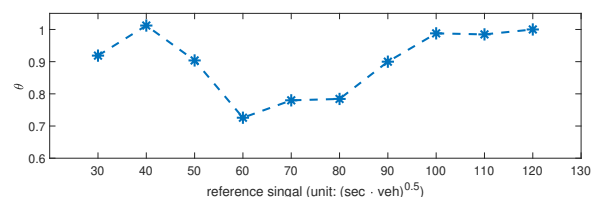

Fig. 13: Tuning parameter for the reference signal

From Fig. 13, the optimal value of y_{ref} is 60, thus this value is chosen for the simulations. Choosing y_{ref} values beyond 120 results in all vehicles being determined as active during the whole simulation period, so those values are excluded from the analysis. For the tuning of K_P and K_I the same procedure is repeated and the parameter values are chosen according to their resulting θ values. Note that this tuning procedure involves choosing the parameters for a specific AMoD operation scenario. More general parameter-tuning methods will be studied in future research.

REFERENCES

[1] G. Zardini, N. Lanzetti, M. Pavone, and E. Frazzoli, "Analysis and control of autonomous mobility-on-demand systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, p. null, 2022. [Online]. Available: https://doi.org/10.1146/annurev-control-042920-012811

[2] C. V. Beojone and N. Geroliminis, "On the inefficiency of ride-sourcing services towards urban congestion," *Transportation Research Part C: Emerging Technologies*, vol. 124, p. 102890, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X20307907

[3] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017. [Online]. Available: https://www.pnas.org/content/114/3/462

[4] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3583–3590.

[5] M. Nourinejad and M. Ramezani, "Ride-sourcing modeling and pricing in non-equilibrium two-sided markets," *Transportation Research Part B: Methodological*, vol. 132, pp. 340–357, 2020, 23rd International Symposium on Transportation and Traffic Theory (ISTTT 23). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261518311500

[6] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing vehicle distributions and fleet sizes for shared mobility-on-demand," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3853–3859.

[7] A. Wallar, M. Van Der Zee, J. Alonso-Mora, and D. Rus, "Vehicle rebalancing for mobility-on-demand systems with ride-sharing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4539–4546.

[8] K. Marczuk, H. Soh, C. L. Azevedo, D.-H. Lee, and E. Frazzoli, "Simulation framework for rebalancing of autonomous mobility on demand systems," 2016.

[9] S. Illgen and M. Höck, "Literature review of the vehicle relocation problem in one-way car sharing networks," *Transportation Research Part B: Methodological*, vol. 120, pp. 193–204, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261518300547

[10] M. Repoux, M. Kaspi, B. Boyacı, and N. Geroliminis, "Dynamic prediction-based relocation policies in one-way station-based carsharing systems with complete journey reservations," *Transportation Research Part B: Methodological*, vol. 130, pp. 82–104, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S019126151930102X

[11] I. A. Forma, T. Raviv, and M. Tzur, "A 3-step math heuristic for the static repositioning problem in bike-sharing systems," *Transportation Research Part B: Methodological*, vol. 71, pp. 230–247, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261514001726

[12] A. Faghih-Imani, R. Hampshire, L. Marla, and N. Eluru, "An empirical analysis of bike sharing usage and rebalancing: Evidence from barcelona and seville," *Transportation Research Part A: Policy and Practice*, vol. 97, pp. 177–191, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856416311648

[13] G. Laporte, F. Meunier, and R. W. Calvo, "Shared mobility systems," *4OR quarterly journal of operations research*, vol. 13, no. 4, pp. 341–360, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10288-015-0301-z

[14] G. H. A. Correia and A. P. Antunes, "Optimization approach to depot location and trip selection in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 233–247, 2012, select Papers from the 19th International Symposium on Transportation and Traffic Theory. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1366554511000822

[15] B. Boyacı, K. G. Zografos, and N. Geroliminis, "An optimization framework for the development of efficient one-way car-sharing systems," *European Journal of Operational Research*, vol. 240, no. 3, pp. 718–733, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221714005864

[16] E. Cepolina and A. Farina, "A methodology for planning a new urban car sharing system with fully automated personal vehicles," *European Transport Research Review*, vol. 6, 06 2013.

[17] A. Kek, R. Cheu, Q. Meng, and C. Fung, "A decision support system for vehicle relocation operations in carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, pp. 149–158, 01 2009.

[18] M. Pavone, K. Treleaven, and E. Frazzoli, "Fundamental performance limits and efficient polices for transportation-on-demand systems," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5622–5629.

[19] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Load balancing for mobility-on-demand systems," in *Robotics: Science and Systems*, 2011.

[20] P. Marco, L. S. Stephen, F. Emilio, and R. Daniela, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012. [Online]. Available: https://doi.org/10.1177/0278364912444766

[21] A. Carron, F. Seccamonte, C. Ruch, E. Frazzoli, and M. N. Zeilinger, "Scalable model predictive control for autonomous mobility-on-demand systems," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 635–644, 2021.

[22] M. Ramezani and M. Nourinejad, "Dynamic modeling and control of taxi services in large-scale urban networks: A macroscopic approach," *Transportation Research Part C: Emerging Technologies*, vol. 94, pp. 203–219, 2018, iSTTT22. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X17302152

[23] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of autonomous mobility-on-demand systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1382–1389.

[24] F. Miao, S. Lin, S. Munir, J. A. Stankovic, H. Huang, D. Zhang, T. He, and G. J. Pappas, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ser. ICCPS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 100–109. [Online]. Available: https://doi.org/10.1145/2735960.2735961

[25] J. Wen, J. Zhao, and P. Jaillet, "Rebalancing shared mobility-on-demand systems: A reinforcement learning approach," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 220–225.

[26] L. Chen, A. H. Valadkhani, and M. Ramezani, "Decentralised cooperative cruising of autonomous ride-sourcing fleets," *Transportation Research Part C: Emerging Technologies*, vol. 131, p. 103336, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21003399

[27] C. Pasquale, S. Sacone, S. Siri, and A. Ferrara, "Hierarchical centralized/decentralized event-triggered control of multiclass traffic networks," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1549–1564, 2021.

[28] M. Mancini, N. Bloise, E. Capello, and E. Punta, "Sliding mode control techniques and artificial potential field for dynamic collision avoidance in rendezvous maneuvers," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 313–318, 2020.

[29] S. Martinez, J. Cortes, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[30] V. Krishnan and S. Martínez, "A multiscale analysis of multi-agent coverage control algorithms," *Automatica*, vol. 145, p. 110516, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109822003752

[31] S. Papatheodorou, Y. Stergiopoulos, and A. Tzes, "Distributed area coverage control with imprecise robot localization," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, 2016, pp. 214–219.

[32] S. Papatheodorou and A. Tzes, *Theoretical and Experimental Collaborative Area Coverage Schemes Using Mobile Agents*, 11 2018.

[33] S. Papatheodorou, A. Tzes, K. Giannousakis, and Y. Stergiopoulos, "Distributed area coverage control with imprecise robot localization: Simulation and experimental studies," *International Journal of Advanced Robotic Systems*, vol. 15, no. 5, p. 1729881418797494, 2018. [Online]. Available: https://doi.org/10.1177/1729881418797494

[34] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[35] A. Carron and M. N. Zeilinger, "Model predictive coverage control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6107–6112, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320322898

[36] I. I. Hussein and D. M. Stipanovic, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[37] Y. Ru and S. Martinez, "Coverage control in constant flow environments based on a mixed energy-time metric," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 112–117.

[38] C. Song and Y. Fan, "Coverage control for mobile sensor networks with limited communication ranges on a circle," *Automatica*, vol. 92, pp. 155–161, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109818301018

[39] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[40] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, pp. 637–676, 1999.

[41] S.-k. Yun and D. Rusy, "Distributed coverage with mobile robots on a graph: Locational optimization," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 634–641.

[42] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, 2012.

[43] M. Erwig, "The graph voronoi diagram with applications," *Networks*, vol. 36, pp. 156–163, 2000.

[44] F. Mohseni, A. Doustmohammadi, and M. B. Menhaj, "Distributed receding horizon coverage control for multiple mobile robots," *IEEE Systems Journal*, vol. 10, no. 1, pp. 198–207, 2016.

[45] C. R. He, J. I. Ge, and G. Orosz, "Fuel efficient connected cruise control for heavy-duty trucks in real traffic," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2474–2481, 2020.

[46] P. Zhu, I. I. Sirmatel, G. F. Trecate, and N. Geroliminis, "Idle-vehicle rebalancing coverage control for ride-sourcing systems," in *2022 European Control Conference (ECC)*, 2022, pp. 1970–1975.

[47] N. P. Rougier, "A density-driven method for the placement of biological cells over two-dimensional manifolds," *Frontiers in Neuroinformatics*, vol. 12, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fninf.2018.00012

[48] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen.:," *Journal für die reine und angewandte Mathematik*, vol. 1909, no. 136, pp. 210–271, 1909. [Online]. Available: https://doi.org/10.1515/crll.1909.136.210

[49] N. Geroliminis and C. F. Daganzo, "Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings," *Transportation Research Part B: Methodological*, vol. 42, no. 9, pp. 759–770, 2008.

[50] A. Zockaie, M. Saberi, and R. Saedi, "A resource allocation problem to estimate network fundamental diagram in heterogeneous networks: Optimal locating of fixed measurement points and sampling of probe trajectories," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 245–262, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X17303340

[51] I. I. Sirmatel and N. Geroliminis, "Nonlinear moving horizon estimation for large-scale urban road networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 4983–4994, 2020.

[52] I. I. Sirmatel, D. Tsitsokas, A. Kouvelas, and N. Geroliminis, "Modeling, estimation, and control in large-scale urban road networks with remaining travel distance dynamics," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103157, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21001753

[53] M. Paipuri, E. Barmpounakis, N. Geroliminis, and L. Leclercq, "Empirical observations of multi-modal network-level models: Insights from the pneuma experiment," *Transportation Research Part C: Emerging Technologies*, vol. 131, p. 103300, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21003090

[54] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, Jun. 1962. [Online]. Available: https://doi.org/10.1145/367766.368168

[55] M. Erwig, "The graph voronoi diagram with applications," *Networks*, vol. 36, pp. 156–163, 10 2000.

[56] H. Yang, X. Liang, Z. Zhang, Y. Liu, and M. M. Abid, "Statistical modeling of quartiles, standard deviation, and buffer time index of optimal tour in traveling salesman problem and implications for travel time reliability," *Transportation Research Record*, vol. 2674, no. 12, pp. 339–347, 2020. [Online]. Available: https://doi.org/10.1177/0361198120954867

[57] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, 2012.

[58] S. G. Lee, Y. Diaz-Mercado, and M. Egerstedt, "Multirobot control using time-varying density functions," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 489–493, 2015.

[59] X. Xu and Y. Diaz-Mercado, "Multi-agent control using coverage over time-varying domains," in *2020 American Control Conference (ACC)*, 2020, pp. 2030–2035.

[60] C. V. Beojone and N. Geroliminis, "Relocation incentives for ride-sourcing drivers with path-oriented revenue forecasting based on a markov chain model," *Transportation Research Part C: Emerging Technologies*, vol. 157, p. 104375, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X23003650

Pengbo Zhu received the B.Sc. degree in Automation in 2017 and M.Sc. degree in Control Science and Engineering in 2019 from Harbin Institute of Technology. Currently, she is a doctoral assistant at the Urban Transport Systems Laboratory (LUTS) of EPFL. Her research interests are the application of control algorithms in urban transportation systems, and the optimization of emerging mobility systems.

Isik Ilber Sirmatel is an assistant professor of automatic control at the Faculty of Engineering, Trakya University. He received the B.Sc. degrees in mechanical and control engineering from Istanbul Technical University, the M.Sc. degree in mechanical engineering from ETH Zurich, and the Ph.D. degree in electrical engineering from EPFL, in 2010, 2012, 2014, and 2020, respectively. Prior to joining Trakya University in 2021, he was a postdoctoral researcher at EPFL. His research focuses on applications of automatic control and optimization, with an emphasis on model predictive control of intelligent transportation systems.

Giancarlo Ferrari-Trecate (SM'12) received the Ph.D. degree in Electronic and Computer Engineering from the Universita' Degli Studi di Pavia in 1999. Since September 2016, he has been Professor at EPFL, Lausanne, Switzerland. In the spring of 1998, he was a Visiting Researcher at the Neural Computing Research Group, University of Birmingham, UK. In the fall of 1998, he joined the Automatic Control Laboratory, ETH, Zurich, Switzerland, as a Postdoctoral Fellow. He was appointed Oberassistent at ETH in 2000. In 2002, he joined INRIA, Rocquencourt, France, as a Research Fellow. From March to October 2005, he was a researcher at the Politecnico di Milano, Italy. From 2005 to August 2016, he was Associate Professor at the Dipartimento di Ingegneria Industriale e dell'Informazione of the Universita' degli Studi di Pavia. His research interests include scalable control, microgrids, machine learning, networked control systems, and hybrid systems. Giancarlo Ferrari Trecate is the founder of the Swiss chapter of the IEEE Control Systems Society and is currently a member of the IFAC Technical Committees on Control Design and Optimal Control. He has been serving on the editorial board of Automatica for nine years and of Nonlinear Analysis: Hybrid Systems.

Nikolas Geroliminis is a full professor at EPFL and the head of the Urban Transport Systems Laboratory (LUTS). He has a diploma in Civil Engineering from the National Technical University of Athens (NTUA) and an M.Sc. and Ph.D. in civil engineering from University of California, Berkeley. He is a member of the Transportation Research Board's Traffic Flow Theory Committee. He also serves as an Associate Editor in Transportation Research, part C, Transportation Science and IEEE Transactions on ITS and in the editorial board of Transportation Research, part B, Journal of ITS and of many international conferences. His research interests focus primarily on urban transportation systems, traffic flow theory and control, public transportation and logistics, on-demand transportation, optimization and large scale networks. He is a recent recipient of the ERC starting grant "METAFERW: Modeling and controlling traffic congestion and propagation in large-scale urban multimodal networks".