Overcoming Mobility Poverty with Shared Autonomous Vehicles: A Learning-Based Optimization Approach for Rotterdam Zuid

Breno Beirigo(✉), Frederik Schulte, and Rudy R. Negenborn

Delft University of Technology, Delft, The Netherlands
{b.alvesbeirigo,f.schulte,r.r.negenborn}@tudelft.nl

Abstract. Residents of cities' most disadvantaged areas face significant barriers to key life activities, such as employment, education, and healthcare, due to the lack of mobility options. Shared autonomous vehicles (SAVs) create an opportunity to overcome this problem. By learning user demand patterns, SAV providers can improve regional service levels by applying anticipatory relocation strategies that take into consideration when and where requests are more likely to appear. The nature of transportation demand, however, invariably creates learning biases towards servicing cities' most affluent and densely populated areas, where alternative mobility choices already abound. As a result, current disadvantaged regions may end up perpetually underserviced, therefore preventing all city residents from enjoying the benefits of autonomous mobility-on-demand (AMoD) systems equally. In this study, we propose an anticipatory rebalancing policy based on an approximate dynamic programming (ADP) formulation that processes historical demand data to estimate value functions of future system states iteratively. We investigate to which extent manipulating cost settings, in terms of subsidies and penalties, can adjust the demand patterns naturally incorporated into value functions to improve service levels of disadvantaged areas. We show for a case study in the city of Rotterdam, The Netherlands, that the proposed method can harness these cost schemes to better cater to users departing from these disadvantaged areas, substantially outperforming myopic and reactive benchmark policies.

Keywords: Mobility poverty · Shared autonomous vehicles · Approximate dynamic programming

1 Introduction

Service levels of residents from different areas of a city can vary significantly due to an uneven distribution of transport resources. Peripheral or low-income

This research is supported by the project "Dynamic Fleet Management (P14-18 – project 3)" (project 14894) of the Netherlands Organization for Scientific Research (NWO), domain Applied and Engineering Sciences (TTW).

E. Lalla-Ruiz et al. (Eds.): ICCL 2020, LNCS 12433, pp. 492–506, 2020.
https://doi.org/10.1007/978-3-030-59747-4_32

regions are typically more prone to suffer from accessibility poverty, that is, the difficulty of reaching certain key activities (e.g., employment, education, healthcare) due to mobility poverty, which is concerned with the systemic lack of transportation and mobility options [10]. Since low-income and mobility poverty are strongly correlated, offering sufficient transportation choices to disadvantaged areas can ultimately improve social equity.

Shared autonomous vehicles (SAVs) and, more generally, autonomous mobility-on-demand (AMoD) systems, offer an opportunity to overcome mobility poverty. Sharing services reduce the cost of personal mobility once all expenses of purchasing, maintaining, and insuring vehicles are distributed across a large user-base [15]. Recent research has demonstrated that efficient SAV fleet management can help AMoD providers fulfilling today's transportation demand using much fewer vehicles. However, typical performance measures fail to account for differences in demographics appropriately, lacking nuanced equity implications [5].

Due to natural demand patterns or deliberate profit-seeking policies, SAVs can end up re-enforcing existing inequalities by frequently moving to regions that are more prone to generate higher profits. Such a preference for affluent regions can already be identified in the current transportation landscape, where mobility options (e.g., ride-hailing, micro-mobility, ride-pooling, and transit) abound in cities' central areas.

In this study, we propose an approximate dynamic programming (ADP) algorithm to schedule and rebalance a fleet of AVs to improve the mobility of targeted disadvantaged areas. This algorithm uses demand data throughout an iterative process to derive value function approximations (VFAs) that convey the expected contribution of system states. These lookahead approximations are then considered in the optimization process to assess the future outcome of current decisions. We illustrate our method using the case of Rotterdam, The Netherlands, where the northern region (Rotterdam Noord) encompasses the entire city center, outperforming the southern (Rotterdam Zuid) in a range of socio-geographical factors, such as income and transport connectivity. To improve the mobility of the residents in the Zuid region and ultimately their access to key activities, we investigate to which extent ride subsidization and rejection penalties can contribute to overall fairness, adequately driving vehicles to underserviced areas.

We consider a first-mile case study in which users request vehicles from a private AMoD provider to access the closest train station (see Fig. 1). This setup is particularly relevant for the deployment of mobility-as-a-service (MaaS) solutions, which are based on the integration of different transport services. We show that a proper cost scheme setup can overcompensate the rebalancing bias towards densely populated and high-income areas improving mobility choices in underserviced areas. Ultimately, our results help city managers to understand the cost of laying out equitable transportation policies that balance providers' profitability and the service levels of underserviced users.

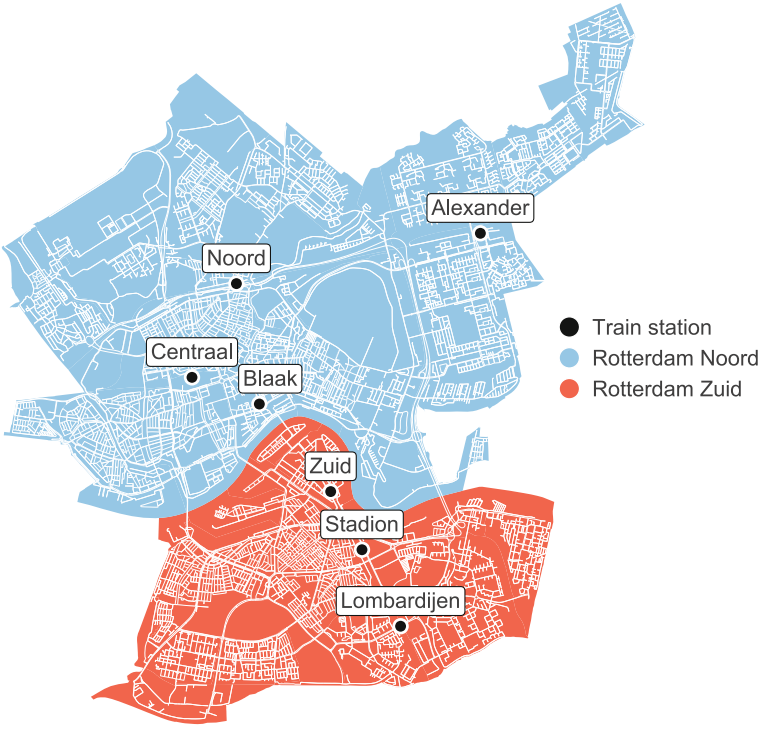

Fig. 1. Rotterdam regions (Noord and Zuid) and the seven train stations that are the destinations of all first-mile trips.

2 Related Work

AMoD systems rely on rebalancing strategies to find a reasonable compromise between asset utilization and user satisfaction. Supply and demand mismatches are typically addressed using ongoing imbalance cues (e.g., request rejections, idle vehicles) and predicted demand information (based on historical data). For example, Pavone et al. [12] propose an optimal transport problem in which locations with a surplus of idle vehicles continuously send empty vehicles to locations with a shortage of idle vehicles. Similarly, Alonso-Mora et al. [2] present a reactive rebalancing approach that sends idle vehicles to undersupplied areas, which are identified by the occurrence of unsatisfied requests. Through a linear program, vehicles are assigned to the departure locations of these requests, aiming to minimize the total sum of travel times. Later, Alonso-Mora et al. [3] use past historical data to compute a probability distribution over future demand and proposes an assignment algorithm to match vehicles to future requests. Fagnant et al. [6] relies on a rule to overcome supply-demand imbalances using a

block-based division operational map. For each block, they compare the supply of idle stationary vehicles versus the share of currently waiting travelers plus soon expected travelers in the near future (according to the block historical trip rate).

Learning-based methods have also been successfully employed to enable anticipatory rebalancing. Through a reinforcement learning (RL) framework, Wen et al. [16] train a deep Q-network (DQN) using rewards based on waiting time savings of users picked up due to rebalancing movements. Conversely, penalties are applied when vehicles remain idle during the rebalancing period. Considering a grid map, they model states using grid-wise idle-vehicle distribution, in-service vehicles, and predicted demand (based on a Poisson process) in the surroundings. Guériau et al. [7] propose a decentralized RL approach based on Q-learning. They show that agents (i.e., vehicles) can contribute to global performance by learning how to optimize their own individual performance with local information only. Lin et al. [9] also takes advantage of the RL framework through a contextual multi-agent actor-critic (cA2C) algorithm. Their design stands out due to two main features, (i) the adoption of centralized value functions (shared by all agents), and (ii) context embedding that establishes explicit coordination among agents. Iglesias et al. [8] design a model predictive control (MPC) algorithm that leverages customer demand forecasts to rebalance vehicles. The forecasting model is based on a long short-term memory (LSTM) neural network. Al-Kanj et al. [1] use an ADP formulation that allows for anticipatory rebalancing and recharging of electric vehicles. Their approach maximizes vehicle contribution over time, using value function approximations to estimate the impact of each decision in the future.

Similarly to [1], we enable anticipatory rebalancing by using value functions to steer vehicles towards high-demand areas. In contrast with all proposed methods, however, we add nuance to service levels, acknowledging that users from different regions face distinct accessibility barriers to the transport system. Based on Lucas et al. [11], we consider transport accessibility primarily in terms of availability and affordability. Additionally, following Cohen et al. [5], our AMoD rebalancing policy aims to complement public transit and redistribute transport resources towards disadvantaged areas. Ultimately, from the user experience perspective, related literature focus on decreasing total delays (pickup and/or in-vehicle) whereas we focus on distributing service levels throughout regions.

3 Problem Formulation

We model the problem using the language of dynamic resource management (see [1,14]), where AVs (resources) service a sequence of trip request batches (tasks) dynamically revealed at discrete-time $t \in \{1,\ 2,\ \ldots,\ T\}$.

We assume all requests arrive in batch intervals of five minutes, occurring within the earliest time t_e = 6:00 and the latest time t_l = 12:00. To ensure the system has enough time to rebalance vehicles and deliver all users, we add thirty-minute offsets before t_e and after t_l, such that the total horizon $T = 84$ (i.e., 420/5).

The state of a single resource is defined by the attribute a representing the vehicle's location in the node-set N of $G = (N, E)$, a strongly connected graph drawn from a section of Rotterdam, The Netherlands. The city comprises six districts and 45 neighborhoods from which we select 40 to exclude the peripheries, such that node and edge sets have sizes $|N| = 10,364$ and $|E| = 23,048$, respectively (see Fig. 1).

By including the temporal dimension to location a, we have a_t, or the location of an SAV at time t. Let $\mathcal{A}$ be the set of all possible vehicle attributes. The state of all vehicles with the same state attribute is modeled using

$$R_{ta} = \text{ Number of vehicles with attribute } a \text{ at time } t,$$
$$R_t = (R_{ta})_{a\in\mathcal{A}} = \text{The resource state vector at time } t.$$

Each request, in turn, is modeled using an attribute vector b comprised of origin and destination attributes $b_1, b_2 \in N$. Let $\mathcal{B}$ be the set of all possible request attribute vectors. The state of all rides with the same state vector occurring at time t is modeled using

$$D_{tb} = \text{ The number of requests with attribute vector } b \text{ at time } t,$$
$$D_t = (D_{tb})_{b\in\mathcal{B}} = \text{The request state vector at time } t.$$

With the resource and request state vectors, we defined our system state vector as $S_t = (R_t,\ D_t)$. States S_t are measured before making decisions at each epoch $t \in \{1,\ 2,\ 3, \ldots, T\}$. In this study, we consider each vehicle can realize three different types of decisions, namely, **service** a single user at a time, **stay parked** in its current location waiting to pick up users, and **rebalance** to a more promising location. Decisions are described using

d^{stay} = Decision to stay parked in the current location,
$\mathcal{D}^{\mathrm{R}}$ = Set of all decisions d to rebalance (i.e., move empty) to a set of neighboring locations,
$\mathcal{D}^{S}$ = Set of all decisions d to service a user,
b_d = Trip $b \in \mathcal{B}$ covered by decision $d \in \mathcal{D}^S$,
$\mathcal{D} = \mathcal{D}^{\mathrm{S}} \cup \mathcal{D}^{\mathrm{R}} \cup d^{stay}$,
x_{tad} = Number of times decision d is applied to a vehicle with attribute a at time t,
$x_t = (x_{tad})_{a\in\mathcal{A}, d\in\mathcal{D}}$.

The decision variables x_{tad} must satisfy the following constraints:

$$\sum_{d\in\mathcal{D}} x_{tad} = R_{ta} \qquad \forall\ a \in \mathcal{A} \tag{1}$$

$$\sum_{a\in\mathcal{A}} x_{tad} \leq D_{tb_d} \qquad \forall\ d \in \mathcal{D}^S \tag{2}$$

$$y_{tb} = D_{tb} - \sum_{a\in\mathcal{A}} \sum_{\substack{d\in\mathcal{D}^S \\ b_d = b}} x_{tad} \qquad \forall b \in \mathcal{B} \tag{3}$$

Constraints (1) and (2) guarantee flow conservation of vehicles and requests, respectively, and equalities (3) define the number y_{tb} of rejected trips b at time t. Once we aim to distinguish regions in N, we define $U(a) : N \rightarrow L$ as a function that maps each location $a \in N$ to a discrete geographical area in L, with $L = \{\text{Noord, Zuid}\}$. Applying a decision d to a resource with attribute a at time t generates a contribution c_{tad}, such that

$$c_{tad} = \begin{cases} p^a_{base} + p_{time}\delta\left(b_1, b_2\right) - c_{time}\left(\delta\left(a, b_1\right) + \delta\left(b_1, b_2\right)\right), & (service), \\ -c_{time}\delta\left(a, r\right), & (rebalance), \\ 0, & (stay). \end{cases}$$

Contributions c_{tad} of service, rebalance, and stay decisions are comprised of

$$\begin{aligned} p^a_{base} &= \text{Base fare of trips departing from region } U(a) \in L, \\ p_{time} &= \text{Time-dependent fare}, \\ c_{time} &= \text{Vehicle time-dependent costs (e.g., fuel)}, \\ c^a_{penalty} &= \text{Penalty for rejecting users from region } U(a) \in L, \\ \delta\left(a, b_1\right) &= \text{Pickup duration}, \\ \delta\left(b_1, b_2\right) &= \text{Trip duration}, \\ \delta\left(a, r\right) &= \text{Rebalance duration (to neighboring location } r). \end{aligned}$$

Assuming contributions are linear, the contribution function for period t discounted by rejection penalties is given by

$$C_t\left(S_t, x_t\right) = \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} c_{tad} x_{tad} - \sum_{b \in B} c^{b_1}_{penalty} y_{tb}. \tag{4}$$

Let $X^\pi_t(S_t)$ be a decision function that represents a policy $\pi \in \Pi$, which maps a state S_t to a decision x_t at time t. We aim to determine the optimal policy π^* that, starting from an initial state S_0, maximizes the expected cumulative contribution, over all the time periods:

$$F^*_0\left(S_0\right) = \max_{\pi \in \Pi} \mathbb{E}\left\{\sum_{t=0}^{T} C_t\left(S_t, X^\pi_t(S_t)\right) | S_0\right\}. \tag{5}$$

4 Algorithmic Strategies

In principle, we can solve Eq. (5) by recursively computing (backward through time) Bellman's optimality equations, assigning to each state S_t at t, a value V_t, such that an optimal policy can chose decisions x_t that maximize expected contributions over time:

$$X^*_t(S_t) = \mathbf{arg} \max_{x_t \in \mathcal{X}_t} \left(C_t\left(S_t, x_t\right) + \mathbb{E}\left\{V_{t+1}\left(S_{t+1}\right) | S_t, x_t\right\}\right). \tag{6}$$

Solving Eq. (6), however, requires computing the expectation, which is computationally intractable for our problem setting. Doing so would incur in all the three "curses of dimensionality" (see [13]), since we would have to enumerate the state, outcome, and decision spaces.

4.1 An Approximate Dynamic Programming Algorithm

Once we cannot determine the true value V_t associated to each state S_t, we use an approximate dynamic programming algorithm (see [13] for a comprehensive treatment) to determine the value $\overline{V}_t^n$, which is an statistical estimate of V_t after n sample observations. First, at each time t in iteration n, applying the decision vector x_t to state S_t^n, before any new information has arrived, leads to a deterministic post-decision state

$$S_t^{x,n} = S^{M,x}(S_t^n, x_t^n),$$

where $S^{M,x}(.)$ is a transition function (or "system model") which describes how the system evolves from S_t^n to $S_t^{x,n}$. As exogenous information is unveiled, we can also use the post-decision state $S_t^{x,n}$ and the transition function to compute the subsequent pre-decision state

$$S_{t+1}^n = S^{M,W}(S_t^{x,n}, W_{t+1}(\omega^n)),$$

where $S^{M,W}(.)$ is a transition function from $S_t^{x,n}$ to S_{t+1}, and $W_{t+1}(\omega^n)$ is the exogenous information integrating a particular set of outcomes $W_1(\omega^n)$, ..., $W_T(\omega^n)$ of the resource and demand vectors measured over all periods in iteration n, following a sample path ω^n .

We solve (6) by replacing the expected value of being in S_{t+1} for $\overline{V}_t^{n-1}(S_t^{x,n})$, which corresponds to an approximation of the value of being in the post-decision state $S_t^{x,n}$ considering the first $n-1$ iterations. Then, we can make decisions at time t by solving the optimization problem

$$F_t\left(S_t^n\right) = \underset{x_t \in \mathcal{X}_t^n}{\max} \left(C_t\left(S_t^n, x_t\right) + \overline{V}_t^{n-1}\left(S_t^{x,n}\right)\right), \tag{7}$$

where we seek to determine the decision vector x_t in the feasible region $\mathcal{X}_t^n$ that maximizes the sum of the current contribution and the pre-calculated expected contribution $\overline{V}^{n-1}$ associated with post-decision state $S_t^{x,n}$.

4.2 Value Function Updates

We use the approximate value iteration algorithm from [13] to update value functions approximations using the solutions of (7) at each period t of iteration n. To streamline the process of stepping forward in time, we assume that the post-decision state is equivalent to the post-decision resource vector. Thus, after decision time, the post-decision demand vector $D_t^{x,n}$ is always empty, such that $\overline{V}_t^n\left(S_t^{x,n}\right) = \overline{V}_t^n\left(R_t^{x,n}\right)$. In practice, this assumption entails that requests are not carried over periods. Consequently, users have to turn to alternative mobility options upon being rejected.

Assuming $\overline{V}_t^n\left(R_t^{x,n}\right)$ is linear in R_{ta}, we have

$$\overline{V}_t^n\left(R_t^{x,n}\right) = \sum_{a' \in \mathcal{A}} \overline{v}_{t'a'}^n \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} \delta_{a'}\left(a, d\right) x_{tad},$$

where $\overline{v}^n_{t'a'}$ is the marginal value of a vehicle with attribute a' at time t' at iteration n. The transition function $\delta_{a'}(a,d)$ is equal to 1 when $a^M(a,d) = a'$, and 0 otherwise, such that a' represents the post-decision location of a, and t' is the arrival time at a' of a vehicle departing from a at time t. When $d = d^{stay}$, $t' = t+1$ and when $d \in \mathcal{D}^S \cup \mathcal{D}^R$, t' accounts for the travel time $\tau(t,a,d)$ to travel from a to a', such that $t' = t + \tau(t,a,d)$.

The marginal values $\overline{v}^n_{ta}$ approximate the overall contribution (i.e., until the end of the simulation horizon T) of assigning an incremental vehicle to a certain location at a certain time. Once costs depend on the region trips depart from (Noord or Zuid), these values will also reflect the benefits of working within these regions.

We update value functions $\overline{v}^n_{ta}$ using the samples $\hat{v}^n_{ta}$ drawn from attribute a at time t and iteration n. New samples are smoothed using stepsizes α_n which are updated every iteration according to the McClain's rule $\alpha_n = \frac{\alpha_{n-1}}{1+\alpha_{n-1}-\overline{\alpha}}$, where $\overline{\alpha}$ is a constant that is approached as n advances. Initially, we set $\alpha_1 = 1$ such that value functions can start with the first sample value measured for each state. Algorithm 1 compiles all the steps of our ADP approach.

Algorithm 1. Approximate dynamic programming

for $n = 1, \ldots, N$ **do**
  Choose a sample path ω^n.
  for $t = 0, 1, \ldots, T$ **do**
    Let x^n_t be the solution of the optimization problem: $x^n_t = F_t(S^n_t)$.
    Let $\hat{v}^n_{ta}$ be the dual corresponding to the resource conservation.
    If $R_{ta} > 0$, update the value function using: $\overline{v}^n_{ta} = (1-\alpha_n)\overline{v}^{n-1}_{ta} + \alpha_n \hat{v}^n_{ta}$
    Update the state: $S^{x,n}_t = S^{M,x}(S^n_t, x^n_t)$ and $S^n_{t+1} = S^{M,W}(S^{x,n}_t, W_{t+1}(\omega^n))$.
  end for
end for
Return the value functions, $\{\overline{v}^n_{ta}, t = 1, \ldots, T, \ a \in \mathcal{A}\}$.

4.3 Hierarchical Aggregation for Value Function Estimation

In order to estimate the value function of state attributes not yet observed, we use hierarchical aggregation coupled with the *weighting by inverse mean squared errors* (WIMSE) formula (see [14]). In this method, the state space is aggregated into a sequence of increasingly coarser state spaces, each of which associated with an aggregation level. By combining the values from superior levels through weights, we can estimate states' value functions without visiting them. We define three hierarchical levels experimentally, namely, 1, 2, and 3, that aggregate states both in space and time.

Spatially, the node set is aggregated in hexagon bins of 0.17 km^2, 0.46 km^2, and 5.16 km^2, resulting in 1,016, 198, and 38 bins, respectively (see Fig. 2). Valid bins cover at least one node of Rotterdam's street network and are identified by

the closest node to their geographical center. As such, we assume that the travel time between two bins is based on the shortest path between their corresponding center nodes at 20 km/h speed. We aggregate temporally by increasing the length of the periods. We assume level 1 is the disaggregate five-minute period, whereas levels 2 and 3 correspond to ten- and fifteen-minute periods, therefore totaling 42, and 38 periods, respectively.

In practice, marginal values will aggregate up to the same value function, for example, at the third level, if they are within the same 5.16 km^2 hexagon and occur throughout the same fifteen-minute bin. Ultimately, the state-space size for each aggregation level declines from 85,344 to 8,316, and then to 1,444.

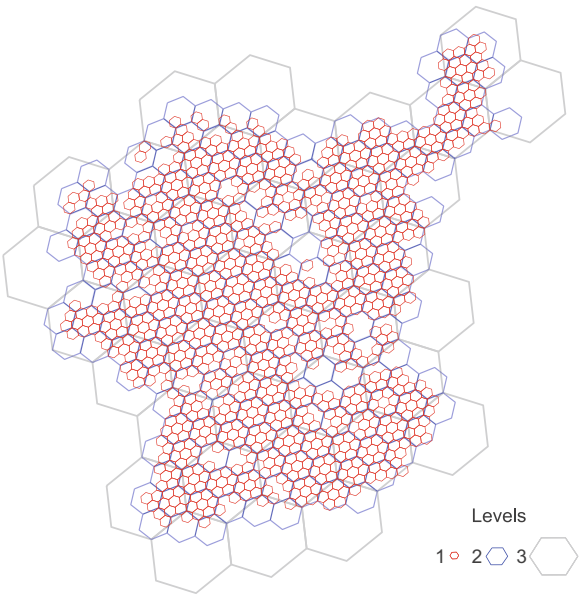

Fig. 2. The three spatial aggregation levels set up for the Rotterdam area encompassing the street network G. Starting from level 1, hexagon bins cover an increasingly higher number of locations of the node-set N.

4.4 Rebalancing Strategies

Slicing the area of network G using a hierarchy of geometric shapes allows us to infer a relation of proximity between nodes within the same region. We exploit this relation by assuming vehicles are allowed to rebalance to the center of the surrounding hexagon neighbors across all three hierarchical levels, totaling up to eighteen rebalancing options. This way, vehicles can explore increasingly farther neighborhoods, insofar as rebalancing targets become hexagon centers up in the spatial hierarchy. To prevent vehicles from flooding high demand locations, we

bound the number of vehicles they can accommodate to v_{max}. Then, we consider that rebalancing trips can take place only when $v_{it} \leq v_{max}$, where v_{max} is the cumulative number of vehicles that are either inbound to or staying at each location $i \in N^{(g)}$ for $g = 1$ from current time period t onward. By doing so, we avoid both methodological and practical problems.

First, although we assume $\overline{V}_t^n(R_t^{x,n})$ is linear in R_{ta}, we acknowledge this assumption is prone to result in an oversupply of vehicles in regions associated with high marginal values. However, instead of dampening these values as the number of vehicles increases (using, for example, piecewise-linear approximations), we prefer to tune the value of constant v_{max}.

Second, from a practical perspective, a real-world application has to account for road capacity and curbside space before rebalancing vehicles. The maximum number of cars across regions can also depend on city regulations and vary according to local restrictions, for example, to avoid inconveniencing residents or businesses of a determined area.

5 Experimental Study

Spatial Demand Patters: To generate r request departure points, we use a weighted random process to select an origin neighborhood according to its relative population density using the Dutch census [4]. Next, we use a regular random process to select a street node within this origin neighborhood. We use this process to select 3,000 request origins, from which about one-third end up within the Zuid region. Figure 3 presents the probability distribution of selecting request origins across Rotterdam neighborhoods. Since we investigate first-mile trips, destination points correspond to the closest train stations of each origin point.

Temporal Demand Patters: Regarding the time the requests arrive at the system, we propose five scenarios in which we vary the demand patterns of residents departing from Noord and Zuid regions. We consider that the number of requests always peaks at 8:00 for the residents of Rotterdam Noord, whereas the number of requests originated in Rotterdam Zuid peak at 6:00, 7:00, 8:00, 9:00, and 10:00, leading to five request arrival scenarios labeled N8Z6, N8Z7, N8Z8, N8Z9, and N8Z10. By varying the demand peaks in Zuid, we can investigate how well the algorithm can distribute vehicles in the light of different levels of competition with the Noord demand. Moreover, the relative position between Noord and Zuid peaks allows us to assess the efficacy to which vehicles can move between regions in anticipation to demand. For instance, the earlier the demand peak in Zuid, the more time vehicles will have to move from Zuid to Noord. Conversely, the later the demand peak, the more time vehicles will have to move from Noord to Zuid. All these scenarios are modeled using a normal distribution truncated by t_e and t_l, with a standard deviation of one hour and means equal to the demand peaks entailed by each region.

Waiting Times: Upon receiving a five-minute request batch, the system sets up available trip decisions taking into account a pickup radius $w_{pk} = 10$ min.

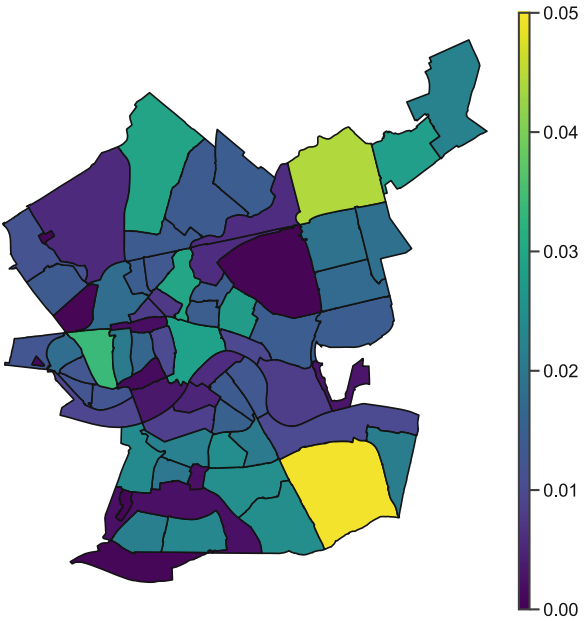

Fig. 3. Probabilities of choosing a departure location within each Rotterdam neighborhood based on the part-to-whole ratio of the number or residents.

If users cannot be accessed by any vehicle within w_{pk}, they are immediately rejected, having to resort to another transportation means. Since we assume the AMoD system integrates a broader MaaS ecosystem, a rejection means that the users will have to rely on alternative modes to fulfill their trips. Hence, in the worst-case scenario, serviced users wait at most fifteen minutes to be picked up.

Cost Schemes: We investigate the influence of six cost schemes on Zuid riders' service levels by manipulating both base fares and penalties. We refer to these schemes using the labels B1R0, B4R0, B8R0, B1R1, B1R4, B1R8, where B and R correspond to the p_{base} and $c_{penalty}$ constants, and the digits represent scaling factors. For instance, B1R0, B4R0, and B8R0 represent cost schemes in which the base fare p_{base} of Zuid users is one, four, and eight times higher. Regarding the values adopted, we consider $p_{base} = c_{penalty} =$€2.5, time-dependent fare $p_{time} =$ 1€/km, and time-dependent operational costs $c_{time} = 0.1$ €/km. Finally, we assume B1R0 is our reference cost scheme and use it for all Noord users.

Fleet Configuration: We determine the fleet size following the model predictive control (MPC) algorithm proposed by [8]. Their approach assumes perfect information throughout the whole horizon and does not allow for delays, such that, at the end of each time step, there is a sufficient number of vehicles to pick up all requests at each location. To decrease computation times, we assume that all trip origins and destinations are associated with their respective third-level hexagon centers (38 in total). We have found that the average fleet sizes

achieved using the optimal MPC formulation across ten demand realizations for each demand pattern scenario range from 382.8 (N8Z10) to 499.9 (N8Z8). The results also indicated that the closer the demand peaks of Noord and Zuid regions are, the higher is the fleet size required. To deliberately create a scarcity scenario that splits the vehicle workforce between the two regions, we carry out all experiments using a 300-SAV fleet. At the beginning of each ADP iteration, we randomly distribute these vehicles throughout level-one hexagon locations. Consequently, since Noord is broader and more populated than Zuid, a service bias towards Noord will naturally emerge.

Benchmark: We benchmark our ADP π_{VFA} policy against two alternative policies, namely, π_{myopic} and $\pi_{reactive}$, in which no information about the future is available. Both policies aim to maximize the cost function in Eq. (4), but while π_{myopic} seeks only to determine the optimal vehicle-request assignment represented by Eqs. (1) and (2), the $\pi_{reactive}$ policy relies on an additional vehicle rebalancing phase. The rebalancing is based on the state-of-the-art algorithm proposed by Alonso-Mora et al. [2], which consists of a linear program where idle vehicles are optimally rebalanced to under-supplied locations. Ultimately, this program aims to minimize the total travel distance of reaching the pickup locations of unassigned requests while guaranteeing that either all vehicles or all requests are assigned.

6 Results

We implemented our approach using Python 3.6 and Gurobi 8.1. Test cases were executed on a 2.60 GHz Intel Core i7 with 32 GB RAM. For all case studies, we run our ADP algorithm throughout 1,000 iterations, considering stepsizes $\overline{\alpha} = 0.1$, and maximum vehicle count $v_{max} = 5$, which have been found to show good performance experimentally.

Since we aim to improve mobility for Zuid users, our analysis focuses mainly on the service levels (i.e., the ratio of serviced requests) achieved for each region. Still, provided that users can be picked up timely, delays are already bound to w_{pk}. Table 1 presents the average service levels (across ten demand realizations) for each policy π and cost scheme configuration. We separate the service levels between users departing from each region to evaluate the effect of the proposed cost schemes on driving vehicles towards Zuid. The results are subsumed under three categories, namely, "Baseline", "Base fare," and "Rejection penalty." The "Baseline" category comprises the averages achieved using the myopic and reactive policies, which we use to benchmark the performance of our learning-based method. As hypothesized, when equity concerns are disregarded, service levels differ markedly between regions, being consistently higher in Noord.

It can be seen from Table 1 that $\pi_{reactive}$ can already significantly improve the service levels of Zuid users, servicing about 10% more requests than π_{myopic} in all demand scenarios. Since the reactive rebalancing policy relies on trip rejection stimuli to move vehicles to undersupplied areas, the fleet is disproportionately

Table 1. Average ratio of serviced users departing from Noord and Zuid regions for each policy, demand scenario, and cost scheme. Figures correspond to the mean average of ten demand distributions over 3,000 requests.

Policy(π)	Cost scheme	Origin in Noord					Origin in Zuid				
		N8Z6	N8Z7	N8Z8	N8Z9	N8Z10	N8Z6	N8Z7	N8Z8	N8Z9	N8Z10
					Baseline						
myopic	B1R0	.783	.788	.795	.790	.795	.637	.668	.688	.694	.682
reactive	B1R0	.807	.846	.850	.852	.853	.778	.775	.783	.790	.773
					Base fare						
VFA	B1R0	.926	.905	.908	.939	.963	.812	.876	.890	.904	.949
	B4R0	.892	.870	.871	.903	.933	.888	.923	.944	.956	.967
	B8R0	.883	.858	.828	.847	.925	.905	.946	.969	.976	.980
					Rejection penalty						
	B1R1	.903	.885	.886	.908	.949	.848	.919	.921	.935	.940
	B1R4	.903	.848	.863	.892	.919	.881	.957	.948	.966	.968
	B1R8	.874	.842	.812	.825	.899	.903	.961	.972	.956	.980

driven to Zuid. However, $\pi_{reactive}$ still leads to a moderate service bias towards the Noord region, regardless of the demand scenario.

We separate the results of our proposed π_{VFA} policy according to the main feature entailed by each cost scheme. Hence, the figures subsumed under the "Base fare" and "Rejection penalty" categories, highlight the effect of scaling up fares and penalties, respectively. It is worth noting that our π_{VFA} policy performs better than the baseline policies, even for cost scheme B1R0, in which no scaling is considered. The performance improvement is especially remarkable when demand peaks from Noord and Zuid are far apart, for example, in scenarios N8Z6 and N8Z10. These scenarios provide enough time for vehicles to rebalance in anticipation from one region to the other instead of reacting to imbalances in short notice. Moreover, during the thirty-minute rebalancing offset previous to the requests' arrival, vehicles also can harness the value functions to reach areas where users are more prone to appear. This explains how even the competitive scenario N8Z8 could benefit from using the π_{VFA} policy.

To investigate the trade-off between profits and Zuid service levels, we average the results of our VFA policy for each cost scheme over all the demand scenarios. Then, we compare the schemes' averages against the averages obtained for the reference cost scheme B1R0. Average profits are determined in terms of base fare accumulation, considering the number of requests departing from each region and the ratios of serviced users in Table 1. Cost schemes B4R0 and B8R0, lead to 104.8% and 249.6% higher profits to service about 4.9% and 6.9% more Zuid users over B1R0. In contrast, cost schemes B1R1, B1R4, and B1R8 incur 3.8%, 9.3%, and 16.7% losses compared to B1R0 average profit, to service about 2.6%, 5.8%, and 6.8% more Zuid users. For both cost scheme categories, the results indicate that further scaling up incentives or penalties is prone to diminishing returns, once less and less service level gains in the Zuid region can be seen.

Throughout all instances considered, scenario N8Z6 has consistently presented the lowest service levels. Since Noord starts with more vehicles than Zuid, and the demand peak in Zuid occurs earlier, this scenario necessarily demands that vehicles move from Noord to Zuid. With most requests happening at 6:00, most rebalancing operations have to be performed within the rebalancing offset. Considering that vehicles can start from remote parts of Noord, the results indicate that the thirty-minute offset is insufficient to rebalance all necessary vehicles to Zuid adequately. As the demand peak in Zuid is pushed forward (e.g., N8Z7), vehicles have more time to access Zuid user origins, and service levels increase.

7 Conclusion

In this study, we present a learning-based fleet rebalancing method that determines a compromise between company revenues and social equity between two distinct regions of Rotterdam, The Netherlands. Our anticipatory rebalancing strategy caters to the needs of targeted regions, compensates for biases towards more affluent and densely populated regions, and mitigates mobility poverty in disadvantaged areas. Based on a range of cost schemes, we show the tipping point at which cost manipulation can affect value function approximations enough to influence the rebalancing process. Ultimately, our approximate dynamic programming algorithm achieves superior service levels compared to myopic and reactive strategies, regardless of cost scheme and demand scenarios considered.

With respect to public polices, our results indicate that the public sector can work in tandem with private providers to guarantee that new mobility solutions consider the patterns of disadvantaged populations. In this way, as mobility technologies develop, private market innovation can be steered to achieve social equity goals, such as preventing mobility poverty. It is worth noting, however, that adequately fulfilling such goals depends on further policy-making. Since the private sector is at the forefront of new mobility systems, such as AMoDs, the public sector has to invest in incentives for the use of these systems in the broader scheme of city transportation. For instance, new services could integrate existing MaaS frameworks, complementing other transportation options (e.g., transit, walking, and cycling).

Future research will focus on designing equity-aware rebalancing strategies to increase user service levels based on alternative information (e.g., age, gender, income), rather than the departure location alone. Additionally, to improve the effectiveness of location-based equity policies, regional transport accessibility can be further investigated, for example, in terms of transit infrastructure, delays, and availability.

References

1. Al-Kanj, L., Nascimento, J., Powell, W.B.: Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. Eur. J. Oper. Res. **284**, 1–40 (2020). https://doi.org/10.1016/j.ejor.2020.01.033

2. Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D.: On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proc. Natl. Acad. Sci. **114**(3), 462–467 (2017). https://doi.org/10.1073/pnas.1611675114
3. Alonso-Mora, J., Wallar, A., Rus, D.: Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3583–3590 (2017). https://doi.org/10.1109/IROS.2017.8206203
4. Centraal Bureau voor de Statistiek (CBS): Wijk-en buurtkaart (2019). https://www.cbs.nl/nl-nl/dossier/nederland-regionaal/geografische-data/wijk-en-buurtkaart-2019
5. Cohen, S., Cabansagan, C.: TransForm: a framework for equity in new mobility (2017). https://trid.trb.org/view/1478022
6. Fagnant, D.J., Kockelman, K.M.: Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas. Transportation **45**(1), 143–158 (2016). https://doi.org/10.1007/s11116-016-9729-z
7. Gueriau, M., Dusparic, I.: SAMoD: shared autonomous mobility-on-demand using decentralized reinforcement learning. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, pp. 1558–1563 (2018). https://doi.org/10.1109/ITSC.2018.8569608
8. Iglesias, R., Rossi, F., Wang, K., Hallac, D., Leskovec, J., Pavone, M.: Data-driven model predictive control of autonomous mobility-on-demand systems. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–7 (2018). https://doi.org/10.1109/ICRA.2018.8460966
9. Lin, K., Zhao, R., Xu, Z., Zhou, J.: Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1774–1783 (2018). https://doi.org/10.1145/3219819.3219993
10. Lucas, K., Mattioli, G., Verlinghieri, E., Guzman, A.: Transport poverty and its adverse social consequences. Proc. Inst. Civ. Eng. - Transp. **169**(6), 353–365 (2016). https://doi.org/10.1680/jtran.15.00073
11. Lucas, K., van Wee, B., Maat, K.: A method to evaluate equitable accessibility: combining ethical theories and accessibility-based approaches. Transportation **43**(3), 473–490 (2015). https://doi.org/10.1007/s11116-015-9585-2
12. Pavone, M., Smith, S.L., Frazzoli, E., Rus, D.: Robotic load balancing for mobility-on-demand systems. Int. J. Robot. Res. **31**(7), 839–854 (2012). https://doi.org/10.1177/0278364912444766
13. Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd edn. Wiley, Hoboken (2011)
14. Simão, H.P., Day, J., George, A.P., Gifford, T., Nienow, J., Powell, W.B.: An approximate dynamic programming algorithm for large-scale fleet management: a case application. Transp. Sci. **43**(2), 178–197 (2009). https://doi.org/10.1287/trsc.1080.0238
15. Spieser, K., Treleaven, K., Zhang, R., Frazzoli, E., Morton, D., Pavone, M.: Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: a case study in Singapore. In: Meyer, G., Beiker, S. (eds.) Road Vehicle Automation. LNM, pp. 229–245. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05990-7_20
16. Wen, J., Zhao, J., Jaillet, P.: Rebalancing shared mobility-on-demand systems: a reinforcement learning approach. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 220–225 (2017). https://doi.org/10.1109/ITSC.2017.8317908