2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

# Vehicle Rebalancing for Mobility-on-Demand Systems with Ride-Sharing

Alex Wallar*, Menno van der Zee†, Javier Alonso-Mora† and Daniela Rus*

*Abstract*— **Recent developments in Mobility-on-Demand (MoD) systems have demonstrated the potential of road vehicles as an efficient mode of urban transportation Newly developed algorithms can compute vehicle routes in real-time for batches of requests and allow for multiple requests to share vehicles. These algorithms have primarily focused on optimally producing vehicle schedules to pick up and drop off requests. The redistribution of idle vehicles to areas of high demand, known as *rebalancing*, on the contrary has received little attention in the context of ride-sharing. In this paper, we present a method to rebalance idle vehicles in a ride-sharing enabled MoD fleet. This method consists of an algorithm to optimally partition the fleet operating area into rebalancing regions, an algorithm to determine a real-time demand estimate for every region using incoming requests, and an algorithm to optimize the assignment of idle vehicles to these rebalancing regions using an integer linear program. Evaluation with historical taxi data from Manhattan shows that we can service 99.8% of taxi requests in Manhattan using 3000 vehicles with an average waiting time of 57.4 seconds and an average in-car delay of 13.7 seconds. Moreover, we can achieve a higher service rate using 2000 vehicles than prior work achieved with 3000. Furthermore, with a fleet of 3000 vehicles, we reduce the average travel delay by 86%, the average waiting time by 37%, and the amount of ignored requests by 95% compared to earlier work at the expense of an increased distance travelled by the fleet.**

## I. Introduction

Autonomous vehicles are enabling a new era of personal mobility with the promise of transportation available anywhere, anytime. Scalable algorithms for motion planning and fleet management that can cope with large request loads are needed for urban deployments. Efficiencies provided by ride-sharing will help optimize those services. In this paper we propose a scalable real-time algorithm for large scale management of fleets of vehicles (autonomous or human-driven), under the ride-sharing model. We build on our previous work [1] and study the role of rebalancing in optimizing a fleet management system.

Ride sharing services such as UberPool and Lyft Line have demonstrated the potential for road vehicles to be used as a sustainable and effective mode of passenger transportation in urban environments. The introduction of e-hailing and global vehicle dispatching in these Mobility-on-Demand (MoD) systems have opened up the opportunity to assign vehicles to requests more efficiently. By allowing multiple passengers to share a single vehicle and considering batches of requests placed around the same time, vehicle routes can be optimized so that less vehicles can serve more requests. This makes the MoD fleet more affordable, sustainable and time-effective, which will be further enhanced by the introduction of autonomous vehicles.

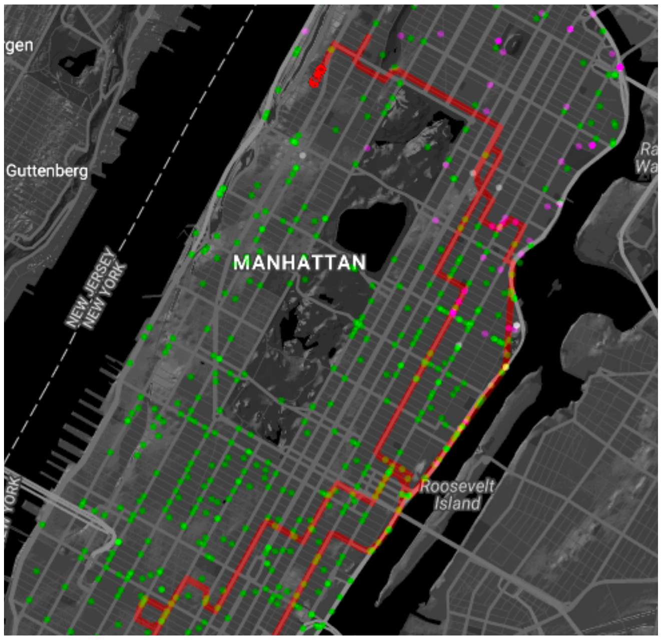


Fig. 1: A snapshot of the simulator. Green dots represent vehicles that either have passengers on board or are on their way to pick up passengers. Pink dots represent vehicles that are rebalancing. Grey dots represent vehicles that are idle. The snapshot furthermore shows the location and history of one of the vehicles in the fleet (represented by the red car).

Besides the computation of efficient vehicle schedules, the proactive relocation of idle vehicles can have a significant influence on the fleet performance. Since the demand for vehicles is often not uniformly distributed, vehicles tend to build up in regions of low demand while vehicles are depleted in regions of high demand. For example in Manhattan, there are many trips to Harlem at night, but fewer back to Midtown in the morning. This mismatch in vehicle supply and demand means that vehicles often have to travel further than necessary to pick up customers, which leads to higher waiting times and more customer walk aways. It also means

* The authors are at the Computer Science and Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, 32 Vassar St, 02139 Cambridge MA, USA {wallar,rus}@csail.mit.edu

† The authors are at the Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628 CD Delft, Netherlands M.J.vanderZee@student.tudelft.nl, J.AlonsoMora@tudelft.nl

*This work was supported in part by the Netherlands Organisation for Scientific Research NWO Veni 15961, the MIT-Singapore Alliance on Research and Technology under the Future of Urban Mobility, and the Office of Naval research grant pDOT ONR N00014-12-1- 1000

that the number of passengers which a fleet can transport in a given time is less than optimal. Vehicle rebalancing focuses on positioning the idle vehicles so that future demand can be served with increased efficiency.

In this work, we build on the vehicle schedule optimization algorithm presented in [1] and expand it with a method to continuously rebalance idle vehicles. We present a method to determine an optimal discretization of the operating area into well defined rebalancing regions and a method to estimate, from incoming transportation requests, a real-time demand per region. Using this estimation we subsequently optimize the rebalancing of idle vehicles towards rebalancing regions. Finally, we present a case study using real taxi data from Manhattan to demonstrate the benefits of our rebalancer and compare it to the previous state of the art.

## A. Related works

The majority of the work on vehicle fleet management for on-demand passenger transportation does not allow for multiple passengers to share a vehicle, focuses on fluid approximations [2], queuing based formulations [3], case studies of specific regions and operational considerations for fleet managers [4]. [5] presented a practical algorithm for assigning vehicle schedules while allowing multiple passengers to share a vehicle. This paper presented a study for New York City which showed that up to 80% of taxi trips in Manhattan could be shared by two riders with a limited increase in travel time.

More recently, an improved algorithm inspired by the work of [5] was presented by [1] for assigning schedules for large numbers of vehicles and requests in real-time while allowing the full capacities of vehicles to be utilised. The work included a naive vehicle rebalancer based on the requests that were rejected from the system due to a lack of available empty seats.

Several works have looked into the redistribution of idle vehicles in a fleet to meet future demand. Most have looked into the redistribution of vehicles in one-way car sharing schemes such as car2go and Zipcar, or bike-sharing schemes. Such systems experience similar mismatches between vehicle supply and demand. Many of these works however focus on infrequent rebalancing (in the order of several times a day) [6], the practical implications of use of human operators to rebalance vehicles [7], or on theoretical formulation and experimentation of the optimization problem [8]. Recent work in [9] shows how vehicle routes can be computed using a heuristic to redistribute bikes in at stations for bike-sharing systems but does directly translate to the MoD paradigm. Prior work has also studied how vehicles can be redistributed to fixed stations by drivers employed by a fleet manager [10]–[12]. However these approaches do not directly apply to the continuous pick up and delivery paradigm of mobility-on-demand.

More relevant to this paper are rebalancing techniques for autonomous mobility-on-demand systems, a car sharing scheme in which vehicles are able to redistribute themselves without the intervention of a human operator. Much of this research shows that a significantly reduced fleet size is required to serve a fixed demand with a similar quality of service [4], [13]. However, these works often use long rebalancing intervals [13] or use simplified models to simulate demand and vehicle movement [4]. The work in [14] presents an model predictive controller to optimize vehicle scheduling and routing, however the experiments were conducted with a relatively small fleet (40 vehicles).

A similar approach to the current work is presented in [15]. However, this approach used predicted demands learned from historical data, which requires elaborate historical data to be available, and presented a naive strategy for rebalancing idle vehicles which assigned idle vehicles to ignored requests.

## B. Contribution

Working further on the earlier work presented in [1] that computes an assignment of a fleet of vehicles to a set of requests, we present a method that continuously rebalances the remaining idle vehicles over the operating area according to estimated real-time demands. Specifically, we present:

- A method to discretize the operating area into a set of rebalancing regions.
- A method to estimate vehicle demand for every rebalancing region using only the real-time request stream.
- An algorithm to assign idle vehicles to rebalancing regions using the estimated demand.
- Experimental validation comparing the performance of using no rebalancer, the rebalancer presented in prior work, and the new proposed rebalancing strategy.

## II. Preliminaries

In this section, all relevant notations used in this paper is presented. Furthermore, we define the problem and present a general overview of the methods employed.

## A. Definitions

We consider a fleet of vehicles $\mathcal{V}$ which can either be autonomous or human-driven. Every vehicle $v \in \mathcal{V}$ has a maximum capacity, $\kappa_v$ and a set of passengers, $P_v$, where $|P_v| \leq \kappa_v$. A passenger is defined as a request that has been picked up by a vehicle and is currently is transit.

We consider set of transportation requests $\mathcal{R} = \{r_1, \ldots, r_n\}$ defined by the tuple $\{o_r, d_r, t_r^r, t_r^{pl}, t_r^*\}$. Here, $o_r$ is the origin, $d_r$ is the destination, $t_r^r$ is the time that the request was placed, $t_r^{pl}$ is the latest acceptable pickup time and $t_r^*$ is the earliest possible time that the destination could be reached. Furthermore, the actual request pick-up time is denoted by $t_r^p$, and the drop-off time is denoted by $t_r^d$.

Vehicles move according to schedules that they are assigned. A schedule $S$ is defined as a sequence of request pick-up and drop-off events, and describes in which order requests are picked up and dropped off.

We define an operating area comprised of a road network as the region the vehicles will consider requests. This operating area is partitioned into a set of rebalancing regions, $\mathcal{G}$, with region centres, $\mathcal{C}$ (described in Sec. III-A). All locations

| | |
|---|---|
| $\mathcal{G}$ | Set of rebalancing regions |
| $\mathcal{C}$ | Set of rebalancing region centers |
| $T_{ij}$ | Travel time from vertex $i$ to $j$ in road-network |
| $\tilde{\lambda}_j$ | Current estimate of rate of requests in region $j$ |
| $\mathcal{V}_r$ | Set of vehicles to rebalance |
| $\mathcal{H}$ | Time horizon for rebalancing |
| $\psi$ | Interval time for batch assignment |
| $\Omega$ | Maximum waiting time |
| $\Delta$ | Maximum travel delay |

TABLE I: Notation used throughout the paper

closer to a region centre $c \in \mathcal{C}$ than any other region centre in $\mathcal{C}$ belong to the associated region.

Additionally, let us define for every request a waiting time $\omega_r = t_r^p - t_r^r$ and a travel delay time $\delta_r = t_r^d - (t_r^r + t_r^*)$. The waiting time and travel delay represent the total amount of time a request waits to be picked up by a vehicle and the time added to the request's transit as opposed to directly travelling to their destination respectively. We also define maximum allowed wait time $\Omega$ and maximum allowed delay time $\Delta$ as in [1]. The values $\Omega$ and $\Delta$ are used as service metrics.

### *B. Problem Formulation*

After every predetermined time interval $\psi$, schedules are computed and assigned to vehicles so that the sum of delays of all requests is minimized. This step will be referred to as the schedule assignment. For a detailed explanation of the algorithm used to assign vehicle schedules, we refer to [1]. In some cases, not all vehicles in the fleet are assigned a schedule. This is either because there are no requests within a travel time smaller than $\Omega$ or because there are other vehicles available that are able to serve the requests more efficiently. Let us denote this set of unassigned vehicles by $\mathcal{V}_r \subseteq \mathcal{V}$. These are the vehicles that are considered for rebalancing. In both cases, there is an oversupply of vehicles in those particular regions. At the same time, other regions might have an under supply of vehicles. In that case, requests in those regions might have to wait significantly longer before they are picked up and eventually might not be able to be serviced while respecting the constraints set by $\Omega$ and $\Delta$.

The focus of this paper is to determine how to distribute the unassigned vehicles over the operating area such that the request delay and waiting time is reduced, the number of ignored requests is minimized, and to do this dynamically over time.

### *C. Method overview*

The method to assign vehicle schedules and rebalance idle vehicles is split up into multiple steps. First, using an integer linear program (ILP), the operating area is discretized into the set of regions $\mathcal{G}$ with region centres $\mathcal{C}$. The regions are computed once offline, and remain constant during the online schedule assignment. This process is explained in Sec. III-A. A schematic overview of the steps performed at every assignment interval $\psi$ is shown in Fig. 2. The following steps are performed:

**Algorithm 1** Overview of the rebalancing method

```
 1: $\mathcal{G} \leftarrow$ DiscretizeOperatingArea($t_{max}$)
 2: for all $g \in \mathcal{G}$ do
 3:     InitializeRateEstimate$_g$()
 4: end for
 5: for every time interval, $\psi$ do
 6:     $\mathcal{R} \leftarrow$ IncomingRequests()
 7:     AssignVehicleSchedules($\mathcal{V}, \mathcal{R}$)
 8:     $Q \leftarrow \{0 : \forall g \in \mathcal{G}\}$
 9:     for all $r \in \mathcal{R}$ do
10:         $g \leftarrow$ GetRebalancingRegion($r$)
11:         $Q_g \leftarrow Q_g + 1$
12:     end for
13:     for all $g \in \mathcal{G}$ do
14:         UpdateRateEstimate$_g$($Q_g, \psi$)
15:     end for
16:     $\mathcal{V}_r \leftarrow$ GetRebalancingVehicles()
17:     RebalanceVehiclesToRegions($\mathcal{V}_r, \mathcal{G}$)
18: end for
```

1) Vehicles are assigned schedules to pick up and drop off requests using the algorithm presented in [1].
2) Using the real-time request information, the current demand at every rebalancing region is estimated using a particle filter. See Sec. III.
3) The vehicles that remained unassigned in the vehicle-schedule assignment (step 1) are assigned to rebalance towards regions in $\mathcal{G}$. This rebalancing assignment is computed using an ILP. See Sec. IV.

All vehicle schedules and rebalancing assignments are reconsidered at every assignment interval. Previously assigned vehicle schedules can change at each subsequent schedule assignment, and vehicles on the way to a rebalancing region can instead be assigned a schedule to pick up passengers. A detailed description of the method overview can be found in Algo. 1.

## III. Demand Estimation

During the execution of the algorithm, we estimate the rate at which incoming requests are being introduced at different locations in our operating area. We do so by first partitioning our operating area into a number of regions and for each region, utilizing a particle filter to estimate the demand.

### *A. Discretization into regions*

Given a directed graph, $G = (V, A)$, representing the road network where $V$ is the set of vertices, and a matrix $T$ where $T_{ij}$ represents the travel time between vertex $i$ and $j$, our problem is to select a subset of vertices $\mathcal{C} \subseteq V$ as region centers that can be used to aggregate demand. A request is in region $g \in \mathcal{G}$ if its origin is closer to the region center $c$ of region $g$ than any other region center in $\mathcal{C}$.

We discretize the operating area into regions using given parameter $t_{max}$ which represents the maximum travel time between any vertex in the graph and the closest region center.

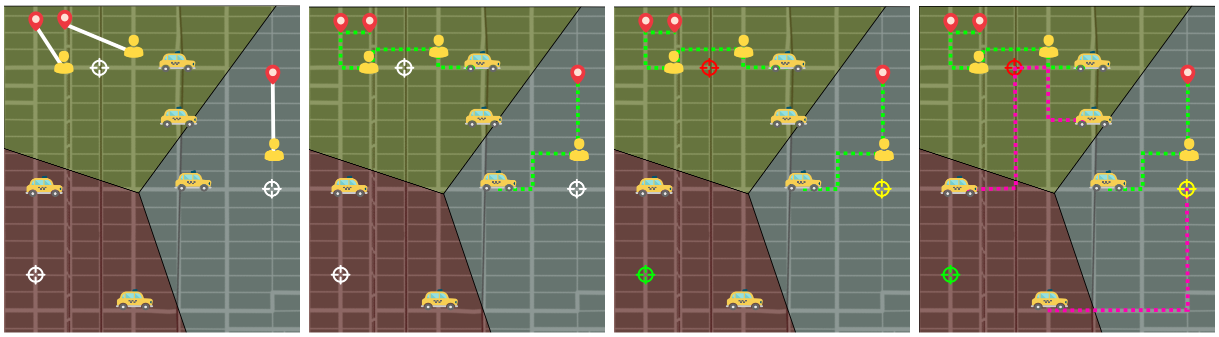

(a) Initial state (b) Schedule assignment (c) Demand estimation (d) Rebalancer assignment

Fig. 2: Schematic overview of the method used for the assignment of vehicles to requests and the rebalancing of vehicles in the order as they are performed. (a) Example for a part of a road network with three regions (white marker = region center), 5 vehicles, and three requests (yellow human = origin, red marker = destination). (b) Assignment of schedules to vehicles, the schedule trajectories are shown by the green dotted lines. Three of the five vehicles are not assigned a schedule. (c) The estimation of demands for every region according to the request information. In this figure, high demand is represented by the red marker, intermediate demand by the yellow marker and low demand by the green marker. (d) Optimized assignment of unassigned vehicles to rebalancing regions. The rebalancing vehicles move towards the region centers. The rebalancing trajectories are shown in the purple dotted lines.

To determine the minimum number of regions for a given $t_{max}$, we formulate the problem as an ILP.

First we define a reachability matrix, $R$, where $R_{ij} = 1$ if $T_{ij} \leq t_{max}$ and $R_{ij} = 0$ otherwise. This describes whether vertex $j$ is reachable from vertex $i$ given the time limit. We also define a set of binary variables $x$ where $x_i = 1$ if vertex $V_i$ is used as a region center and 0 otherwise. Using the reachability matrix and the binary variables, we can define an ILP to determine the minimum number of region centers such that every vertex in the graph is reachable from at least one region center as:

$$\min_{x} \sum_{i=1}^{|V|} x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{|V|} x_i \cdot R_{ij} \geq 1 \quad \forall j \in [1, |V|] \tag{2}$$

Eq. (2) ensures that every node in the road network graph is reachable within $t_{max}$ travel time by at least one region center selected from the nodes in the graph. To extract the region centers, we select from $V$ all vertices $V_i$ such that $x_i = 1$.

The region centers are computed a priori and are used to aggregate requests together so the rate of requests for each region can be computed. These region centers are also used for rebalancing as they are the locations that vehicles are proactively sent to.

## B. Determining the rate of requests

We estimate the vehicle demand online in each rebalancing region using only the real-time request stream. We define the vehicle demand as the rate at which requests are originating from a given region over time.

The rate of requests at each region $g \in \mathcal{G}$ is modelled as an inhomogeneous Poisson point process with a stochastic time-varying rate, $\lambda_g(t)$. These rates are assumed to drift over a short time horizon according to a Wiener process. This means that for each region $g$, the change in rate of requests over time follows a Gaussian distribution, i.e. $\lambda_g(t') - \lambda_g(t) \sim \mathcal{N}(0, \nu \cdot (t' - t))$ for $t' > t$ and some given volatility parameter, $\nu$. The rate, $\lambda_g(t)$, for each region is estimated using a sequential importance resampling particle filter as described in [16]. particle filter is updated with the number of requests observed, $n$, within a time interval, $t - \epsilon_t$ to $t$. The $N$ particles, $\{\hat{\lambda}_0^{(i)} : 1 \leq i \leq N\}$, are initialized uniformly at random within an given upper and lower bound at time 0. Their weights, $\{w_0^{(i)} : 1 \leq i \leq N\}$, are all set to $1/N$. The particles are updated in four steps.

1) $N$ samples, $\{\hat{\lambda}_{t-\epsilon_t}^{(i)}\}$ with weights, $\{w_{t-\epsilon_t}^{(i)}\}$, are drawn with replacement from the particles with probabilities proportional to their weights.
2) Random noise is applied to each particle according to the Wiener process: $\hat{\lambda}_t^{(i)} = \hat{\lambda}_{t-\epsilon_t}^{(i)} + \epsilon_\lambda$, where $\epsilon_\lambda \sim \mathcal{N}(0, \nu \cdot \epsilon_t)$
3) The weights are updated with the observation of $n$ requests in $\epsilon_t$ time: $\tilde{w}_t^{(i)} = w_{t-\epsilon_t}^{(i)} \cdot \Pr[k = n; \epsilon_t \cdot \hat{\lambda}_t^{(i)}]$, where $\Pr[k = n; \epsilon_t \cdot \hat{\lambda}_t^{(i)}]$ is the Poisson probability of $n$ events with a rate $\epsilon_t \cdot \hat{\lambda}_t^{(i)}$
4) The weights are normalized: $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_k \tilde{w}_t^{(k)}}$

The estimate of the stochastic rate, $\lambda_g(t)$, for a region $g$ at time $t$ is then defined as the weighted average of the particles, $\lambda_g(t) = \sum_i w_t^{(i)} \cdot \hat{\lambda}_t^{(i)}$. The particle filter produces an estimate of the rate of requests for a given region by estimating the likelihood of a fixed number of candidate rates and returning the likelihood weighted average over the candidate rates.

## IV. Rebalancing

Due to the fact that demand is not equally distributed over the operating area, vehicles will tend to build up according to a spatial distribution that does not match the distribution of demand. Due to this undesirable distribution of vehicles, it is possible that some vehicles remain idle while there are requests that are not served. This takes places when there are no vehicles that can reach these requests within the maximum waiting time $\Omega$, or when the demand in a specific region is very high and there are not enough vehicles to serve all the requests in that specific region. In order to mitigate this problem, idle vehicles should be proactively rebalanced over the operation area so that their distribution matches the distribution of demand. This furthermore decreases waiting time since for incoming requests, the probability of having a nearby vehicle available is higher. We propose a novel vehicle rebalancer that models the problem as an ILP to match the supply of vehicles to each area with the demand.

### A. Implementation

Our rebalancer seeks to match the supply of idle vehicles in each region to the expected demand for a given time horizon $\mathcal{H}$. Let us define, $\mathcal{V}_r \subseteq \mathcal{V}$ as the set of idle vehicles that are available for rebalancing. These are the vehicles that are not assigned to pick up or drop off requests in the vehicle schedule assignment. Let us also define $\mathcal{C}$ as the set of region centres as described in Sec. III-A. Our goal is to find an assignment from vehicles in $\mathcal{V}_r$ to region centres in $\mathcal{C}$ such that we maximise the amount of requests the vehicles are able to serve, while not over saturating or under saturating regions with vehicles.

To solve this assignment, we can formulate the problem as an ILP. Let us first define a set of binary variables, $\mathcal{X} = \{x_{ij} : \forall i \in [1, |\mathcal{V}_r|], \forall j \in [1, |\mathcal{C}|]\}$, where $x_{ij} = 1$ if vehicle $i$ is assigned to rebalance to region centre $j$ and zero otherwise. Let us also define a travel time matrix, $T$, where $T_{ij}$ is the travel time from vehicle $i$ to the region centre $j$ and a rate vector $\tilde{\lambda}$ where $\tilde{\lambda}_i$ is the current rate of requests at region $i$ computed using the particle filter described in Sec. III-B. With these variables, we can define the objective function for our ILP which we seek to maximize as:

$$\mathcal{J}(\mathcal{X}) = \sum_{i=1}^{|\mathcal{V}_r|} \sum_{j=1}^{|\mathcal{C}|} x_{ij} \cdot \tilde{\lambda}_j \cdot (\mathcal{H} - T_{ij}) \tag{3}$$

This objective represents the sum of the expected number of requests each vehicle would observe in its assigned rebalancing region for the given time horizon, $\mathcal{H}$. The expected number of requests observed by vehicle $i$ is expressed as the rate of requests at the assigned region, $\tilde{\lambda}_j$, multiplied by the time remaining in the time horizon after the vehicle reaches the region, $\mathcal{H} - T_{ij}$

A valid rebalancing assignment must guarantee that each vehicle is assigned to at most one station. This is described in the constraint:

$$\sum_{j=1}^{|\mathcal{C}|} x_{ij} \leq 1 \quad \forall i \in [1, |\mathcal{V}_r|] \tag{4}$$

Also, due to our formulation, we constrain the solution to assign vehicles to rebalancing regions that are reachable within the time horizon, $\mathcal{H}$, i.e. $\mathcal{H} \geq T_{ij}$. This constraint is formulated as:

$$\begin{aligned} x_{ij} \cdot (\mathcal{H} - T_{ij}) &\geq 0 \\ \forall i \in &[1, |\mathcal{V}_r|] \text{ and } j \in [1, |\mathcal{C}|] \end{aligned} \tag{5}$$

In order to obtain an adequate dispersion of vehicles and limit the oversaturation of vehicles in rebalancing regions, we need to constrain the assignment such that the supply of vehicles in a rebalancing region is less than some factor of their demand. The supply of vehicles in region $j \in [1, |\mathcal{C}|]$ for a given time horizon can be written as:

$$\sum_{i=1}^{|\mathcal{V}_r|} x_{ij} \cdot \frac{\mathcal{H} - T_{ij}}{\mathcal{H}} \tag{6}$$

The supply of vehicles is weighted by the percent of time in the next time horizon a vehicle would be able to sit idle at the assigned station. The time weighting is used to give a more accurate estimation of the vehicle supply. For example, if a vehicle takes 8 minutes to reach a region and the time horizon is set to 10 minutes, that vehicle's supply is only available for 20% of the time.

The demand for vehicles for some region $j \in [1, |\mathcal{C}|]$ and a given time horizon is defined as:

$$\tilde{\lambda}_j \cdot \mathcal{H} \tag{7}$$

Putting Eq. (6) and (7) together we formulate a constraint to limit the oversaturation of vehicles in rebalancing regions as:

$$\sum_{i=1}^{|\mathcal{V}_r|} x_{ij} \cdot (\mathcal{H} - T_{ij}) \leq \tilde{\lambda}_j \cdot \mathcal{H}^2 \cdot \rho \quad \forall j \in [1, |\mathcal{C}|] \tag{8}$$

Note that for a more concise description, the time horizon, $\mathcal{H}$, was multiplied on both sides of the inequality. Also note that we have introduced a tuning parameter, $\rho$, that allows us to specify an acceptable level of oversaturation at a rebalancing region.

Combining the objective function from Eq. (3), $\mathcal{J}(\mathcal{X})$, with the constraints described in this section, we formulate an ILP that finds an assignment of vehicles to rebalancing regions that maximizes the expected number of requests observed by all vehicles while obtaining an adequate dispersion of vehicles to limit the oversaturation of vehicles in rebalancing regions. This ILP is then:

$$\begin{aligned} \max_{\mathcal{X}} \quad & \mathcal{J}(\mathcal{X}) \\ \text{s.t.} \quad & \text{constraints } (4), (5), (8) \end{aligned} \tag{9}$$

This optimization will be executed repeatedly after every time interval $\psi$, after vehicles have been assigned schedules to pick up and drop off requests.

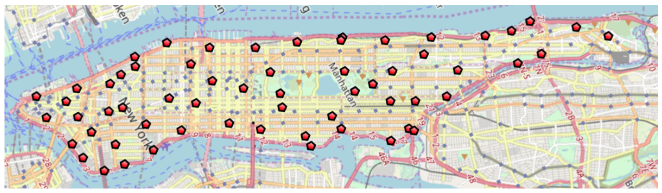


Fig. 3: The computed location of region centers in Manhattan using the algorithm presented in Sec. III for a maximum reachability time $t_{max} = 150$ seconds.

## V. Evaluation

We evaluate the proposed informed rebalancer using historical taxi request data from Manhattan [17] and compare the performance to the state of the art. Since the rebalancing and scheduling algorithms use timeouts to prematurely exit from the optimization, a more powerful computer can lead to much better results. To ensure a fair comparison to previous work, we reimplemented the rebalancer described in [1] and ran experiments on the same machine using the proposed informed rebalancer, the naive rebalancer from [1], without rebalancing and compared the performance of the MoD fleet.

### A. Experimental Setup

For the experiments we use one day of historical taxi data from 00:00 to 23:59 on May $1^{st}$, 2013. This data is publicly available and contains all taxi trips in Manhattan [17]. The data contains the origin, destination, and associated pick up and drop off time for each taxi trip. From this raw data, we use the reported pick up time as the request time since the request time was not provided. The experiments are executed using a simulator that simulates the movement of the vehicles, and to which requests are added according to the historical taxi data. The vehicle routes and travel times are determined using a stored road network of Manhattan. Like [1], we estimate the travel time for each road segment using daily mean travel time computed by the method in [5] and pre-compute the shortest path for every pair of nodes in the road network. A snapshot of the visualizer for this simulator is shown in Fig 1. A computer with a 2.6 GHz (overclocked to 4.0GHz), 18 core (36 threads) Intel i9 processor and 128GB of memory was used to run the experiments.

We assess the performance of the rebalancing algorithm with a fleet size of 1000, 2000, and 3000 vehicles and a capacity of four passengers. We used a fixed maximum waiting time of $\Omega = 3$ minutes and maximum delay of $\Delta = 6$ minutes. All requests that cannot be served within these defined constraints on waiting time and delay time are ignored, and dropped from the request pool. We used 100 particles to estimate the rate of requests in each region. The vehicle locations are initialized uniformly on vertices in the road network. The assignment interval was chosen as $\psi = 30$ seconds as in [1]. This means that vehicle schedules and the assignment of rebalancing stations are optimized every 30 seconds. At assignment time, all requests are considered that have not yet been picked up. To discretize the operating area into rebalancing regions, we used a maximum reachability time of $t_{max} = 150$ seconds which produced 61 regions. The centers of these regions are shown in Fig. 3.

We evaluate two rebalancing techniques: our proposed informed rebalancing algorithm and the naive rebalancing algorithm presented in [1]. The rebalancing algorithm in [1] assigns an idle vehicle to move to the locations of unassigned requests. The assignment minimizes the sum of the distances travelled by the vehicles. We compare the results of these rebalancing techniques to a case were no rebalancing is performed.

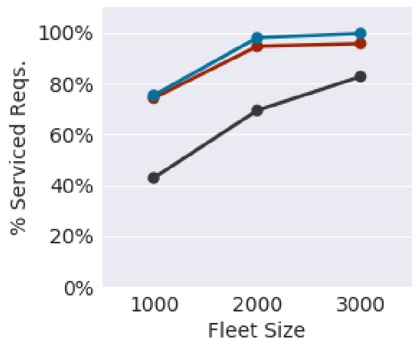
### B. Results

We collect several metrics to assess the performance of the rebalancers including the service rate, in-car travel delay, waiting time, number of ignored requests, distance travelled per vehicle, fleet utilization, and computation time. The service rate is defined as the percentage of the total number of requests that were successfully served within the waiting time and delay time constraints. The in-car travel delay for a request is defined as $\delta_r - \omega_r$. The fleet utilization is defined as the average percent of the fleet with assigned schedules throughout the day. The computational time includes the time required to compute schedules, estimate demands, and compute the rebalancing assignment. These metrics are plotted in Fig. 4. The associated raw data is shown in Tab. II.

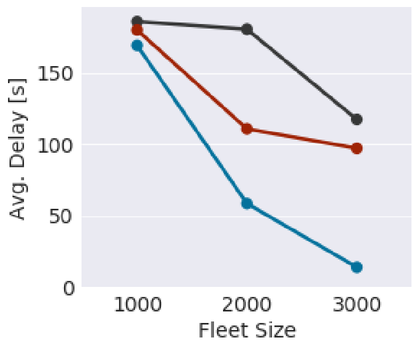
We observe that the service rate improves for all fleet sizes when using the proposed informed rebalancer rather than the naive rebalancer (See Fig. 4a). Most notably, for a fleet size of 3000 vehicles, the service rate increases by 4%. Also the proposed rebalancer achieves a higher service rate with a fleet size of 2000 vehicles (98.1%) than the naive rebalancer with a fleet size of 3000 vehicles (95.8%). This means that by switching to our rebalancing algorithm, you can reduce the size of your fleet by over 33% while maintaining the same service rate. We also see a drastic reduction in the number of requests the algorithm is not able to satisfy for all fleet sizes (See Fig. 4d). In particular, for a fleet size of 3000 vehicles, the proposed algorithm reduces the number of ignored requests by 95% compared to the naive approach.

The in-car travel delay and waiting time also benefit from informed rebalancing (See Fig. 4b and 4c). For all fleet sizes, the average in-car travel delay and waiting time decreases when using the proposed rebalancer. For 3000 vehicles, the average delay drops from 97.1 to 13.7 seconds (86% improvement) and the average waiting time drops from 91.4 to 57.9 seconds (38% improvement).

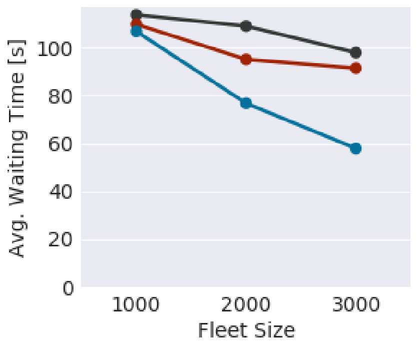
We also observe higher vehicle utilization for the informed and naive rebalancers compared to not rebalancing for all fleet sizes (See Fig. 5). The informed rebalancer achieves the highest vehicle utilization for all fleet sizes. The naive and informed rebalancers achieve similar utilization for a 1000 vehicle fleet, but for 2000 and 3000 vehicle fleets, the informed rebalancer performs much better. This can be explained by the fact that both the naive and informed rebalancer for a 1000 vehicle fleet utilize almost all vehicles continuously over the duration of the experiment.



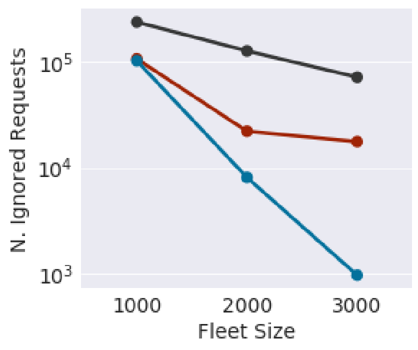
(a) Service Rate



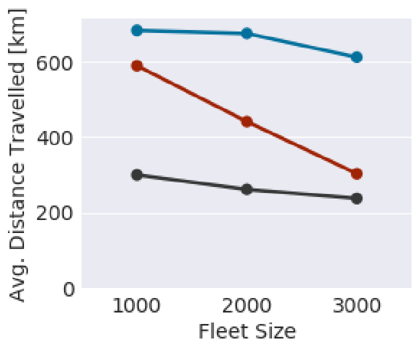
(b) Avg. in-car travel delay



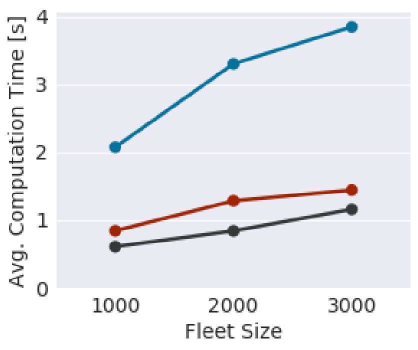
(c) Avg. waiting time



(d) Number of ignored requests



(e) Avg. distance travelled per vehicle



(f) Computation time

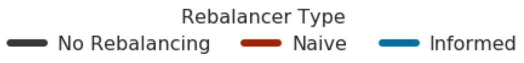


Fig. 4: A comparison of several performance metrics for experiments with a fleet of 1000, 2000 and 3000 vehicles and no rebalancing, naive and informed rebalancing.

| Fleet Size | Rebalancer | Avg. Delay [s] | Avg. Wait Time [s] | Avg. Dist. [km] | Avg. Comp. Time [s] | N. Ignored | % Serviced Reqs. |
|---|---|---|---|---|---|---|---|
| 1000 | No Rebalancing | 185.94 | 113.70 | 300.14 | 0.61 | 239154 | 42.83 |
| 1000 | Naive | 180.03 | 109.76 | 590.02 | 0.84 | 107462 | 74.31 |
| 1000 | Informed | 169.20 | 106.81 | 683.75 | 2.08 | 103022 | 75.37 |
| 2000 | No Rebalancing | 180.53 | 109.00 | 261.39 | 0.84 | 127796 | 69.45 |
| 2000 | Naive | 110.68 | 95.09 | 441.57 | 1.29 | 22196 | 94.69 |
| 2000 | Informed | 58.36 | 76.74 | 675.55 | 3.31 | 8108 | 98.06 |
| 3000 | No Rebalancing | 117.32 | 97.93 | 237.72 | 1.16 | 72048 | 82.78 |
| 3000 | Naive | 97.11 | 91.40 | 303.24 | 1.44 | 17669 | 95.78 |
| 3000 | Informed | 13.72 | 57.85 | 612.56 | 3.85 | 964 | 99.77 |

TABLE II: A detailed overview of the performance metrics for 1000, 2000 and 3000 vehicles for experiments with no rebalancer, and for the informed and naive rebalancer

Our vehicle-trip assignment and rebalancing is efficient enough for online computation. Fig. 4f shows the computational time for different fleet sizes. This time includes computing vehicle schedules, solving the vehicle-trip assignment, and solving the rebalancing assignment for each batch of requests. With a fleet size of 3000 vehicles, our algorithm takes on average less than 4 seconds for a pool-time of 30 seconds which is acceptable for online use.

As in [1] and [15], we observe that the advantages by using a rebalancer come at the cost of an increased distance travelled by the vehicles. This is apparent from figure Fig. 4e. This might lead to higher fuel consumption, but the initial vehicle costs and costs of potential human drivers are much lower when using a smaller fleet with comparable performance. The reason for the larger travel distances is partly because more vehicles are being rebalanced and are moving when they are not assigned. This is also enforced however by the fact that the cost function used for assignment prefers using as many vehicles as possible with an as low as possible occupancy rate when feasible to serve requests to minimize the delay. This cost does not take into account the collective distance travelled by the vehicles.

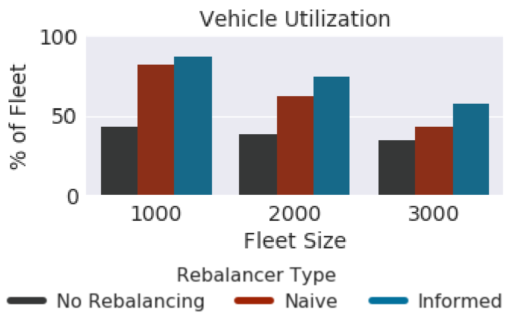


Fig. 5: Vehicle utilization for 1000, 2000, and 3000 vehicle fleet sizes. The vehicle utilization is defined as the average percent of vehicles assigned to trips over the entire experiment. For each fleet size, the vehicle utilization is measured without rebalancing, using the naive rebalancer, and using the proposed informed rebalancer

## VI. Conclusion

In this paper, we presented a method to rebalance idle vehicles in a mobility-on-demand fleet. We presented a method to partition the operating area into a set of rebalancing regions, a method to compute filtered demand estimates for each region based on real-time request information, and a method to optimize the assignment of idle vehicles to rebalancing regions using these demand estimates. Our rebalancing algorithm can be applied to human-driven or autonomous vehicle fleets. We used this rebalancer to significantly improve the efficiency of the ride-sharing algorithm presented in [1] at the expense of longer distance travelled by the vehicles.

We demonstrated a significant improvement in fleet performance using the proposed informed rebalancing strategy over previous work. For a fleet of 3000 vehicles, we reduce the average waiting time by 37%, the travel delay by 86%, and the number of ignored requests by 95% and increase the total distance travelled by 102%.

Future work will focus on developing a method to solve for the assignment of trips to vehicles and idle vehicles to rebalancing regions in a single optimization procedure and to reduce the distance travelled by the vehicles. We also plan to expand the ride-sharing algorithm to utilize fleets with varying vehicle capacities and incorporate the existing public transportation infrastructure.

## References

[1] 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.

[2] M. Pavone, S. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[3] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: a queueing-theoretical perspective," *Proceedings of Robotics: Science and Systems Conference*, July 2014.

[4] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli, "Shared-vehicle mobility-on-demand systems: a fleet operators guide to rebalancing empty vehicles," in *Transportation Research Board 95th Annual Meeting*, no. 16-5987, 2016.

[5] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks.," *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13290–4, 2014.

[6] S. Weikl and K. Bogenberger, "Relocation strategies and algorithms for free-floating car sharing systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 100–111, 2013.

[7] M. Nourinejad, S. Zhu, S. Bahrami, and M. J. Roorda, "Vehicle relocation and staff rebalancing in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 81, pp. 98–113, 2015.

[8] D. Chemla, F. Meunier, and R. Wolfler Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

[9] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve, "Inventory rebalancing and vehicle routing in bike sharing systems," *European Journal of Operational Research*, vol. 257, no. 3, pp. 992 – 1004, 2017.

[10] S. L. Smith, M. Pavone, M. Schwager, E. Frazzoli, and D. Rus, "Rebalancing the rebalancers: Optimally routing vehicles and drivers in mobility-on-demand systems," in *American Control Conference (ACC), 2013*, pp. 2362–2367, IEEE, 2013.

[11] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[12] R. Zhang, F. Rossi, and M. Pavone, "Analysis, control, and evaluation of mobility-on-demand systems: a queueing theoretical approach," *IEEE Transactions on Control of Network Systems*, 2018.

[13] K. A. Marczuk, H. S. Soh, C. M. Azevedo, D.-H. Lee, and E. Frazzoli, "Simulation framework for rebalancing of autonomous mobility on demand systems," in *MATEC Web of Conferences*, vol. 81, p. 01005, EDP Sciences, 2016.

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

[15] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive Routing for Autonomous Mobility-on-Demand Systems with Ride-Sharing," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3583–3590, 2017.

[16] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, Feb 2002.

[17] B. Donovan and D. B. Work, "New York City Taxi Trip Data (2010-2013)." http://dx.doi.org/10.13012/J8PN93H8, 2014.