2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Predictive Routing for Autonomous Mobility-on-Demand Systems with Ride-Sharing

Javier Alonso-Mora*,†, Alex Wallar* and Daniela Rus*

Abstract— **Ride-sharing, or carpooling, systems with autonomous vehicles will provide efficient and reliable urban mobility on demand. In this work we present a method for dynamic vehicle routing that leverages historical data to improve the performance of a network of self-driving taxis. In particular, we describe a constrained optimization method capable of assigning requests to autonomous vehicles in an informed way, to minimize the expected cost of serving both current and future travel requests. We allow several passengers with independent trips to share a vehicle and allow vehicles to pick additional passengers as they progress through their route. Based on historical data, we compute a probability distribution over future demand. Then, samples from the learned probability distribution are incorporated into a decoupled vehicle routing and passenger assignment method to take into account the predicted future demand. This method consists of three steps, namely pruning of feasible trips, assignment of trips to vehicles and rebalancing of idle vehicles. We show the benefits and trade-offs of this predictive approach in an experimental evaluation with over three million rides extracted from a dataset of taxi trips in New York City. Our method produces routes and assignments that, in expectation, reduce the travel and waiting times for passengers, with respect to a purely reactive approach. Besides the mobility on demand application, the method we present is general and could also be applied to other multi-task multi-vehicle assignment and routing problems.**

I. Introduction

Ride sharing services, such as UberPool and Lyft Line, are transforming urban mobility. Also know as vehicle pooling options, these systems allow several passengers, typically limited to two, to share a vehicle when traveling along similar routes. Similar services include Via, which provides vehicle pooling with vans, and Bridj, which provides an alternative to buses. Currently these companies relay on drivers to operate the vehicles, but there is a push in the industry towards autonomous self-driving vehicles. Examples include Google, Uber, Nutonomy and other major car manufacturers. These fleets of autonomous vehicles are expected to provide safe, reliable and affordable transportation.

Efficient algorithms capable of assigning travel requests to a fleet of vehicles, and routing the vehicles efficiently, are required. In this work, we present a constrained optimization method which accounts for future, predicted, requests to route the vehicles. Based on historical data, we first describe a method to compute a probability distribution over future demand. Then, we describe a method for vehicle routing and passenger assignment that takes into account the future demand to produce routes and assignments that, in expectation, reduce the travel and waiting times. Our method can assign thousands of requests to thousands of autonomous vehicles in real time, where we allow that several passengers with independent trips share a vehicle and that a vehicle picks additional passengers as it progresses in its route. It further refines the quality of the assignment over time.

This work could also be applied to other problems which require routing of large fleets of mobile robots to satisfy a set of tasks, i.e. the multi-task multi-robot problem.

A. Related works

Informed driving is becoming a key feature to increase the sustainability of taxi companies and with a combination of readily available large datasets and powerful data mining tools, estimation of future patterns from data is an active field of research. Several works have looked at estimating future demand given past taxi data, for example, [1], [2] and [3]. We adopt a frequentist approach thanks to the large data available and focus on its integration in the assignment and routing problem.

Much of the fleet management literature for mobility-on-demand systems consider the case of ride-sharing without pooling requests, focusing on fluid approximations [4], queuing based formulations [5], case studies in specific regions (e.g., Singapore [6]) and operational considerations for fleet managers [7]. With the growing interest and rapid developments in autonomous vehicles, there has also been an increasing focus on autonomous mobility-on-demand systems [4], [7], [8]. However, none of these works consider the ride-pooling problem of servicing multiple rides with a single trip, nor the prediction of future requests. A study in New York City showed that up to 80% of the taxi trips in Manhattan could be shared by two riders with an increase in the travel time of a couple minutes [9] and also showed the gains attainable by a "global oracle" with full knowledge of the future. These results were confirmed by [10], which introduced an algorithm for real-time request matching and vehicle routing in low and high capacity vehicles.

We build on the constrained optimization approach of [10], which only took into account the state of the fleet and the requests at the current time instance. In this paper we incorporate a prediction of future demand, which aims

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

† The author is currently at the Delft Center for Systems and Control of the Delft University of Technology, Mekelweg 2, 2628 CD Delft, Netherlands {jalonsom,wallar,rus}@csail.mit.edu

*This work was supported in part by pDOT ONR N00014-12-1-1000 and the MIT-Singapore Alliance on Research and Technology under the Future of Urban Mobility

at better positioning the fleet of vehicles towards satisfying future requests.

B. Contribution

We present an efficient constrained optimization method for vehicle routing and multi-request multi-vehicle assignment that takes into account a prediction of the future demands.

In Section III we describe a method to predict future requests based on historical data from taxi trips. These predicted requests are then incorporated in the main algorithm.

The main contribution is a method, described in Section IV, which takes into account the predicted future demand to influence both the vehicle routing and the assignment of requests to vehicles. The method works in the context of ride sharing, and extends the planning horizon beyond the current requests.

Finally, in Section V, we present experimental results with over three million real trip requests extracted from the New York City taxi dataset [11].

II. Preliminaries

In this section we introduce the notation employed throughout this paper, followed by the problem formulation and an overview of the method.

A. Definitions

We consider a fleet $\mathcal{V}$ of m vehicles of capacity ν, the maximum number of passengers each vehicle can have at any given time. Denote the set of vehicles $\mathcal{V} = \{v_1, \ldots, v_m\}$. The current state of a *vehicle* v is given by a tuple $\{q_v, t_v, \mathcal{P}_v\}$, indicating its current position q_v, the current time t_v and its passengers $\mathcal{P}_v = \{p_1, \ldots, p_{n_v^{pass}}\}$. A *passenger* p is a request that has been picked-up by a vehicle.

We also consider a set of requests $\mathcal{R} = \{r_1, \ldots, r_n\}$. Where each travel request consists of the time of request, a pick-up location and a drop-off location. Formally, a *request* r is defined by a tuple $\{o_r, d_r, t_r^r, t_r^{pl}, t_r^*\}$, indicating its origin o_r, its destination d_r, the time of the request t_r^r, the latest acceptable pick-up time t_r^{pl} (initially given by $t_r^{pl} = t_r^r + \Omega$ with Ω the *maximum waiting* time), and the earliest possible time at which the destination could be reached $t_r^* = t_r^r + \tau(o_r, d_r)$. The pick-up time is denoted by t_r^p and the expected drop off time t_r^d.

Given a graph of the streets with estimated travel times, a function $\tau(q_1, q_2)$ computes the travel time from q_1 to q_2, two positions in space encoded by their latitude and longitude coordinates. When a network representation of the map is available, standard techniques for efficiently computing shortest paths can be used [12].

We further define a *trip* $T = \{r_1, \ldots, r_{n_T}\}$ as a set of requests that can be combined and served by a single vehicle. A trip may have one or more candidate vehicles for execution and contain more requests than the capacity of the vehicle if they are picked-up and dropped-off in a way that the capacity limit is satisfied at all times.

B. Problem formulation

We define the following problem.

Problem 1 (Informed batch assignment): Consider a set of requests $\mathcal{R}$, a set of vehicles $\mathcal{V}$ at their current state including passengers, and a function to compute travel times on the road network. Compute the optimal assignment Σ of requests to vehicles that satisfies a set of constraints $\mathcal{Z}$, including a maximum capacity ν of passengers per vehicle, and that minimizes a cost function $\mathcal{C} = \mathcal{C}_{now} + \mathcal{C}_{future}$, where $\mathcal{C}_{now}$ could be the sum of travel delays for the current passengers and requests and $\mathcal{C}_{future}$ is a term which includes the cost of satisfying future predicted travel requests.

Our formulation follows [10] and is flexible with respect to physical and performance-related constraints $\mathcal{Z}$. In our implementation we consider the following ones:

- For each request r, the waiting time ω_r, given by the difference between the pick-up time t_r^p and the request time t_r^r, must be below a maximum waiting time Ω, for example 5 minutes.
- For each request r (or passenger p) the total travel delay $\delta_r = t_r^d - t_r^*$ ($\delta_p = t_p^d - t_p^*$) must be lower than a maximum travel delay Δ, for example 10 minutes, where t_r^d is the drop-off time and $t_r^* = t_r^r + \tau(o_r, d_r)$ the earliest possible time at which the destination could be reached if the shortest path between the origin o_r and the destination d_r was followed without any waiting time. The total travel delay δ_r includes both the in-vehicle delay and the waiting time.
- For each vehicle v, a maximum number of passengers, $n_v^{pass} \leq \nu$, for example capacity four.

Ideally, all the requests shall be assigned to a vehicle, but given the constraints, this might not always be the case. Denote by $\mathcal{R}_{ok}$ the set of requests assigned to a vehicle and $\mathcal{R}_{ko}$ the set of requests that are not served by any vehicle.

Following [10], we define the cost $\mathcal{C}_{now}$ of an assignment Σ as the sum of travel delay over all passengers $\mathcal{P}$ and all assigned requests plus a large enough cost c_{ko} for each non-assigned request. Formally,

$$\mathcal{C}_{now}(\Sigma) = \sum_{p \in \mathcal{P}} (t_p^d - t_p^*) + \sum_{r \in \mathcal{R}_{ok}} (t_r^d - t_r^*) + \sum_{r \in \mathcal{R}_{ko}} c_{ko} \quad (1)$$

To account for the future performance of the system, we introduce a new term $\mathcal{C}_{future}$, which is the expected cost of serving future requests. This cost term is based on the predicted future demand with the objective of achieving a better routing and assignment of the fleet towards the future requests. This will be discussed in Section IV.

For real-time fleet management, the method can be applied to continuous discovery and assignment of incoming requests. The proposed approach is to perform batch assignment of the requests within a short time span, for example every 30 seconds, to the fleet of vehicles. Problem 1 is invoked with the predicted state of the fleet at the assignment time and the cumulated requests. Requests that have not been picked-up by a vehicle within the previous assignment round are kept in the pool for assignment.

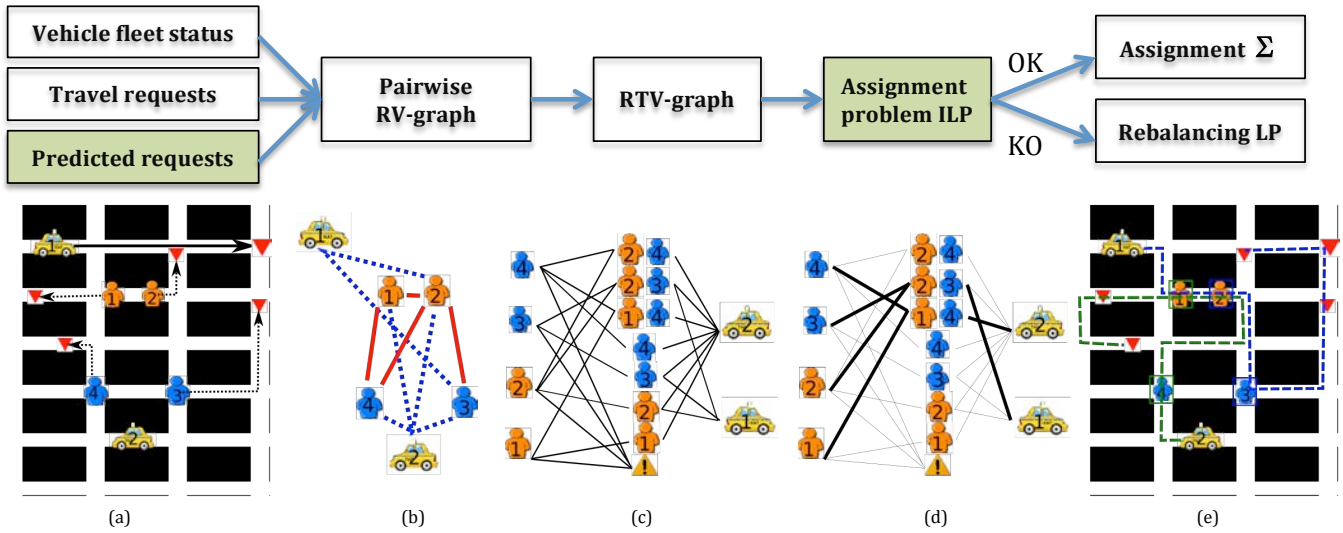


Fig. 1. Schematic overview of the proposed method for batch assignment of multiple requests to multiple vehicles of capacity ν. The method consists of several steps leading to an integer linear optimization which provides an assignment that can be refined over time. The main differences with respect to [10] are the addition of predicted future requests and a modified formulation of the ILP assignment. (a) Example of a street network with two requests (orange human = origin, red triangle = destination), two predicted requests (blue human = origin, red triangle = destination) and two vehicles (yellow car = origin, red triangle = destination of a passenger). Vehicle 1 has one passenger and vehicle 2 is empty. (b) Pairwise shareability RV-graph of requests and vehicles. Cliques of this graph are potential trips. (c) RTV-graph of candidate trips and vehicles which can execute them. A node (yellow triangle) is added for requests that can not be satisfied. (d) Optimized assignment given by the solution of the ILP, where vehicle 1 serves requests 2 and 3 and vehicle 2 serves requests 1 and 4. (e) Planned route for the two vehicles and their assigned requests. The predicted requests alter the route of the vehicles, driving them towards areas of likely future requests.

C. Method overview

The first step of the method consists of estimating, for each time of the day and for each day of the week the amount of requests from each origin in the city to each destination. This is a probability distribution that is computed from historical data. We describe this step in Sec. III.

The main step of the method consists of solving Problem 1. To do so, at each assignment round we sample future requests from the estimated probability distribution and introduce them in the assignment and routing problem, albeit with lower cost that the real requests. This is described in Sec. IV. Fig. 1 shows a schema with the steps of the method, which we describe in the following.

The assignment and routing method is inspired by [10] and consists of the following four steps.

- Computing a pair-wise request-vehicle shareability graph (RV-graph). In this graph, requests r, predicted requests r^{pred} and vehicles v are pairwise connected if r, or r^{pred}, can be satisfied by v within the defined constraints and given the current state of v.
- Computing a graph (RTV-graph) of feasible trips (formed by one or more requests and/or predicted requests) and the vehicles that can serve them within the specified constraints.
- Solving an Integer Linear Program (ILP) to compute the best assignment of vehicles to trips.
- Rebalancing the remaining idle vehicles towards areas with a deficit of vehicles and too many requests via a Linear Program (LP).

Given that the problem at hand is NP hard, obtaining an optimal assignment can be computationally expensive. For practical applications it is required that a sub-optimal solution is returned within an allocated runtime budget, which might be improved incrementally up to optimality. The proposed algorithm does present this anytime-optimal property in the sense that the assignment is refined over time. Yet, to find the truly optimal assignment, a potentially infinite amount of future samples and requests would be required.

III. Prediction of Future Demand

In a preprocessing step, the probability distribution of origin-destination requests is computed for fixed intervals of the week. We do so by discretizing the area into regions and cumulating requests from a year of historical taxi data.

A. Estimation of historical demand

Using a list of all intersections, we discretize the area into regions given by a distance parameter r which relates to the acceptable distance a person would need to walk. With this discretization, we can then assign the origin and destination of the requests from the historical data to the closest region centers. Using a frequentist approach, for each 15 minute interval of the week, we count the number of requests going from every origin region to every destination region. With this frequency table we are able to determine the probability of a given destination region given the origin region, time interval, and day of the week.

1) Discretization into regions: Given a list C_0 of all the intersections in the road network of the city, we compute the set C of region centers such that in the resulting list no two centers are within a given radius r of each other, i.e. $\forall i, j \in C, ||i - j|| > r$. We do this incrementally with Algorithm 1, where BALLTREE is a space partitioning data structure that allows for fast radius bounded nearest neighbor lookup. The data structure has a query function QUERY(c, r), that returns all the points within r of a point c. In Fig. 2 we show the centers of the regions from a discretization of Manhattan, where a radius of 150 meters was used.

Algorithm 1 Incremental Pruning of region centers

$C = C_0$; $\quad i = 0$;
$\mathcal{T} \leftarrow$ BALLTREE(C_0)
while $i < |C|$ **do**
  $c_i = C[i]$ # i-th element in C
  $C \leftarrow C \setminus \mathcal{T}$.QUERY$(c_i, r)$
  $i = i + 1$
end while

2) Probability distribution: Given the set of region centers, we construct a probability distribution, $P(p, d \,|\, \xi, w)$, which is the probability of a request appearing in the origin region p and with destination region d, given the , time interval ξ, and day of the week w. We partition each day into 15 minute intervals resulting in $1 \leq \xi \leq 96$.

This probability distribution can be generated via a frequentist approach. We used one year of historical taxi data consisting of 165,114,362 trips [13]. Each trip contained the origin and destination coordinates along with the time and date of the pick up. Using this data, we were able to populate a $96 \times 7 \times |C| \times |C|$ table, $\mathcal{F}$, indexed by the time interval, day of the week, origin region, and destination region with the number of times a given trip occurred. This allows us to determine the probability of a destination given the origin and a time period. The time period is defined as an initial and final time interval and the day of the week, $I = (\xi_0, \xi_1, w)$, resulting in the probability distribution of origin-destination

$$P(d, p \,|\, I) = P(p \,|\, I) \cdot P(d \,|\, p, I) \tag{2}$$

where

$$P(p \,|\, I) = \frac{\sum_{\xi=\xi_0}^{\xi_1} \sum_{i=1}^{|C|} \mathcal{F}_{\xi,w,p,i}}{\sum_{\xi=\xi_0}^{\xi_1} \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} \mathcal{F}_{\xi,w,i,j}} \tag{3}$$

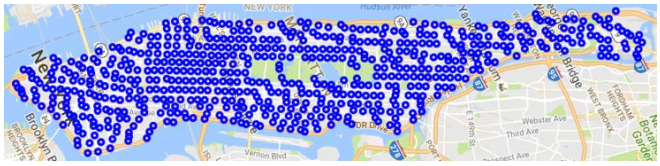


Fig. 2. Region centers determined by the greedy station algorithm that were used in our experiments

and

$$P(d \,|\, p, I) = \frac{\sum_{\xi=\xi_0}^{\xi_1} \mathcal{F}_{\xi,w,p,d}}{\sum_{\xi=\xi_0}^{\xi_1} \sum_{i=1}^{|C|} \mathcal{F}_{\xi,w,p,i}} \tag{4}$$

In Fig. 3 we show an example of the predicted demand for two fixed origins and two different time periods. From this probability distribution we can sample future requests to anticipate demand.

B. Sampling of future demand

Consider a given period of time $\xi = (\xi_0, \xi_1)$ and a day of the week w, and recall that $I = [\xi, w]$. We construct a list S, consisting of the cumulative sum of frequencies from the start time to the end time and another list L, of the same size, consisting of the corresponding origin-destination pairs. To sample requests we then generate a random number s, from 0 to $\max(S)$ and determine the index i of S such that $S_{i-1} \leq s \leq S_i$. We then return L_i which is the corresponding origin-destination pair of the cumulative sum of frequencies interval. This process, see Algorithm 2, allows us to draw samples from $\mathcal{D}(I) \sim P(p, d \,|\, I)$. The function RAND$(0, N)$ returns a uniformly distributed random number from 0 to N. FINDINTERVAL(S, s) returns the index i such that $S_{i-1} \leq s \leq S_i$ if $s > S_0$, otherwise it returns 0. This is done using binary search since S is sorted.

Algorithm 2 Sampling origin-destination pairs over time

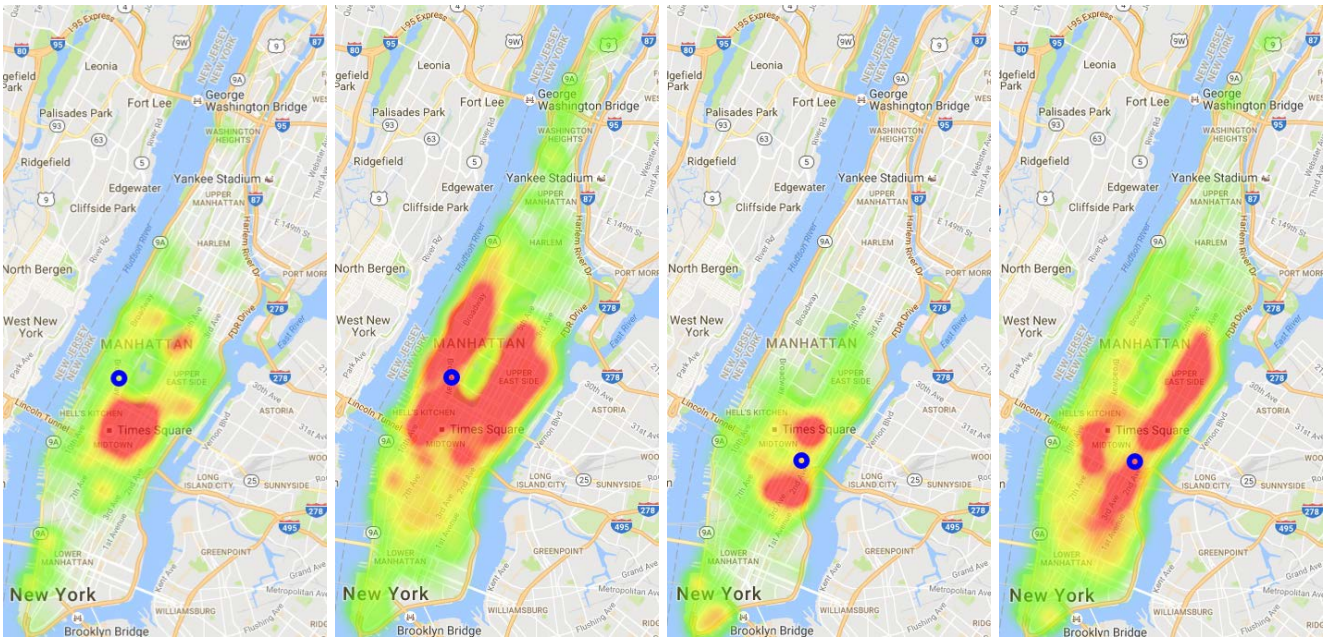
$S \leftarrow \{\}, L \leftarrow \{\}$
for $\xi \in [\xi_0, \xi_1]$ **do**
  for $(p, d) \in [1, |C|]^2$ **do**
    $S \leftarrow S \cup \{\max(S) + \mathcal{F}_{\xi,w,p,d}\}$
    $L \leftarrow L \cup \{(p, d)\}$
  end for
end for
$s \leftarrow$ RAND$(0, \max(S))$
$i \leftarrow$ FINDINTERVAL(S, s)
return L_i

IV. METHOD FOR ROUTING AND ASSIGNMENT

The goal is to bias the vehicles towards areas where future requests are more likely to appear. The method takes into account the current state of the fleet, the current set of requests, as well as the predicted demand, consisting of both origins and destinations. The method computes a batch assignment of the current requests in the requests pool $\mathcal{R}$ to the vehicles of the fleet $\mathcal{V}$. For real scenarios with incoming requests, this routing and assignment is performed at a constant frequency, which in our experiments was once every 30 seconds. Fig. 1 shows a schema with the steps of the method, which we describe in the following.

A. Sampling of future requests

In each iteration of the batch optimizer, a set of additional requests $\mathcal{R}_{future}$ are sampled from a historical probability distribution of future demand with the method described in



(a) Westside, 7:30 – 8:00 (b) Westside, 21:30 – 22:00 (c) Eastside, 7:30 – 8:00 (d) Eastside, 21:30 – 22:00

Fig. 3. Heatmaps depicting the destination demand distribution. For this example, two locations in Manhattan are used as origins with a 30 minute interval to show the distribution. For each location, two intervals are used to show different snapshots of the demand throughout the day.

Sec. III-B. We first define a time interval for the predictions $I_{pred} = [t_{now}, t_{pred}, w]$, where t_{now} is the current time and t_{pred} a time in the future, which in our experiments is set to $t_{now} + 1800s$ for an interval of 30 minutes in the future, and w is the day of the week. We also define a maximum number of samples n^{max}_{pred}.

At run time, the number of samples is given by

$$n_{pred} = \min(n^{max}_{pred}, E(\mathcal{D}(I_{pred}))), \tag{5}$$

where $E(\mathcal{D}(I_{pred}))$ denotes the number of expected requests in interval I_{pred}, given the distribution $\mathcal{D}$ estimated in Sec. III.

Each future request $r^{pred}_i \in \mathcal{R}_{future}$ is sampled, via Algorithm 2, from $\mathcal{D}$ and the time interval,

$$r^{pred}_i \sim \mathcal{D}(I_{pred}). \tag{6}$$

At each time step, after each batch assignment, the set $\mathcal{R}_{future}$ is cleared. New future requests will be sampled in the following time step, every 30 seconds in our experiments.

B. Optimization

These requests are added to the pool of requests $\mathcal{R}^+ := \mathcal{R} \cup \mathcal{R}_{future}$ for the current iteration (and removed afterwards). Vehicles can then be matched with trips containing future requests in $\mathcal{R}_{future}$ and may make progress towards them (although they can not be picked since they are virtual).

The additional requests $\mathcal{R}_{future}$ are subject to the same constraints $\mathcal{Z}$ as the real requests $\mathcal{R}$, and enter the assignment problem via the additional term in the optimization cost $\mathcal{C}_{future}$. Following Eq. (1), this term is defined as

$$\mathcal{C}_{future}(\Sigma) = \sum_{r \in \mathcal{R}^{pred}_{ok}} (t^d_r - t^*_r) + \sum_{r \in \mathcal{R}^{pred}_{ko}} c^{pred}_{ko}, \tag{7}$$

where $\mathcal{R}^{pred}_{ok}$ is the set of assigned future requests and $\mathcal{R}^{pred}_{ko}$ the set of unassigned future requests, such that $\mathcal{R}^{pred}_{ok} \cup \mathcal{R}^{pred}_{ko} = \mathcal{R}_{future}$. The cost of a future request being ignored satisfies $c^{pred}_{ko} << c_{ko}$, much lower than that of real requests. This process gives preference to real requests, with a bias in the assignment and routing towards servicing areas of higher expected future demand.

Following Sec. II-C the batch assignment algorithm consists of the following steps:

- Sample a set of requests $\mathcal{R}_{future} \sim \mathcal{D}$.
- Compute a pair-wise request-vehicle shareability graph (RV-graph) between the requests $\mathcal{R}^+$ and the vehicles $\mathcal{V}$. In this graph, request r and vehicle v are connected if, given the current state of v, request r can be satisfied by v while respecting the defined constraints $\mathcal{Z}$ for maximum waiting time, delay and vehicle capacity.
- Compute a graph (RTV-graph) of feasible trips (formed by one or more requests) and the vehicles that can serve them within the specified constraints. Each trip may contain both real and predicted requests. Feasible trips are computed incrementally for each vehicle. Each trip is linked in the graph to the requests that form it and the vehicles that can serve it while respecting the constraints $\mathcal{Z}$.
- Compute a greedy assignment Σ_{greedy}, where trips are assigned to vehicles iteratively in decreasing size of the trip and increasing cost. The idea is the maximize the amount of requests served while minimizing cost.
- Starting from the greedy assignment solve an Integer Linear Program to compute an optimal assignment Σ_{optim} of vehicles to trips, and therefore to requests,

Algorithm 3 Optimal assignment

1: Initial guess: Σ_{greedy}
2: $\Sigma_{optim} := \arg\min_{\mathcal{X}} \sum_{i,j \in \mathcal{E}_{TV}} c_{i,j}\epsilon_{i,j} +$
3: $\quad + \sum_{\forall r_k \in \mathcal{R}} c_{ko}\chi_k + \sum_{\forall r_k \in \mathcal{R}_{future}} c_{ko}^{pred}\chi_k$
4: $\quad$ s.t. $\sum_{i \in \mathcal{I}^T_{V=j}} \epsilon_{i,j} \leq 1 \qquad \forall v_j \in \mathcal{V}$
5: $\quad \sum_{i \in \mathcal{I}^T_{R=k}} \sum_{j \in \mathcal{I}^V_{T=i}} \epsilon_{i,j} + \chi_k = 1 \qquad \forall r_k \in \mathcal{R}^+$

for the cost function. Following [10], a binary variable is added for each link between a feasible trip and a vehicle that can execute it within the RTV-graph. This assignment also provides the optimal routes, as computed in the RTV-graph.

- Rebalance the remaining idle vehicles towards areas with a deficit of vehicles and too many requests via a Linear Program. The idle vehicles are assigned to the unassigned requests of the previous step.

Following the notation of [10], the new Integer Linear Program (fifth step of the method, see Algorithm 3) consists of the following binary variables

$$\mathcal{X} = \{\epsilon_{i,j},\ \chi_k;\ \forall (T_i, v_j) \text{ edge in RTV-graph}, \forall r_k \in \mathcal{R}^+\}.$$

From Eq. (1) and Eq. (7) the cost terms $c_{i,j}$ are given by the sum of delays for all the passengers and requests associated to a trip T_i, as served by a vehicle v_j

$$c_{i,j} = \sum_{r \in \mathcal{I}^R_{T=i,V=j}} (t^d_r - t^*_r) + \sum_{p \in \mathcal{I}^P_{V=j}} (t^d_p - t^*_p), \qquad (8)$$

where $\mathcal{I}^R_{T=i,V=j}$ denotes the requests in trip T_i as served by vehicle v_j, and $\mathcal{I}^P_{V=j}$ the passengers of vehicle v_j.

The optimal assignment is obtained by solving the ILP of Algorithm 3. Recall that: $\mathcal{E}_{TV}$ denotes the edges between a trip T_i and a vehicle v_j in the RTV-graph (i.e. there exists a route for which vehicle v_j can serve trip T_i within the given constraints $\mathcal{Z}$); $\mathcal{I}^T_{V=j}$ denotes the trips that can be served by vehicle v_j; $\mathcal{I}^T_{R=k}$ denotes the trips (combinations of requests) in which request r_k can be served; and $\mathcal{I}^V_{T=i}$ denotes the vehicles that can serve trip T_i.

After assignment and routing, the vehicles make progress towards their assigned requests, picking requests (which become passengers) as they reach them, and the set $\mathcal{R}_{future}$ is cleared. Then, this process is repeated at the desired frequency with the incoming requests.

C. Properties

1) Complexity.: The number of variables in the ILP is equal to the number of edges $e(T, v)$ in the RTV graph plus the number of requests in $\mathcal{R}^+$. In the worst case, it is of order $\mathcal{O}(m(n + n_{pred})^\nu)$, only reached with complete RV- and RTV-graphs where all vehicles can serve all requests and all requests can be combined with each other. In practice, the number of variables is orders of magnitudes lower and related to the size of the cliques in the RV-graph, but does scale poorly with the number of predicted requests n_{pred}, since they can typically be combined with many of the current requests (since they are at a future time at which some of the passengers might have been dropped off). The number of constraints is $n + n_{pred} + m$.

2) Anytime optimality.: The proposed model includes the constraints $\mathcal{Z}$ and the cost term $\mathcal{C} = \mathcal{C}_{now} + \mathcal{C}_{future}$ which aim at minimizing the total delay in expectation, with respect to the current passengers, the current requests and the expected future demand $\mathcal{D}$. With respect to the model, and the sampled requests, this method guarantees optimality of the assignment, while satisfying the constraints $\mathcal{Z}$, if all the steps are executed until termination and exploration of all possible trips and assignments. A potentially infinite amount of samples might be required to achieve optimality - in expectation - with respect to the original cost function. In practice, time-outs are set both for the amount of time spent generating candidate trips for each vehicle, and for the amount of time spent exploring the branches of the ILP. A limit on the number of vehicles considered per request, the number of trips per vehicle or the optimality gap of the ILP can also be set. These timeouts trade-off optimality for tractability and their values will depend on the available resources. In contrast to [10], the algorithm is not anymore reactive but does take into account a prediction of the future demand. The method seamlessly allows for parallelization in all steps.

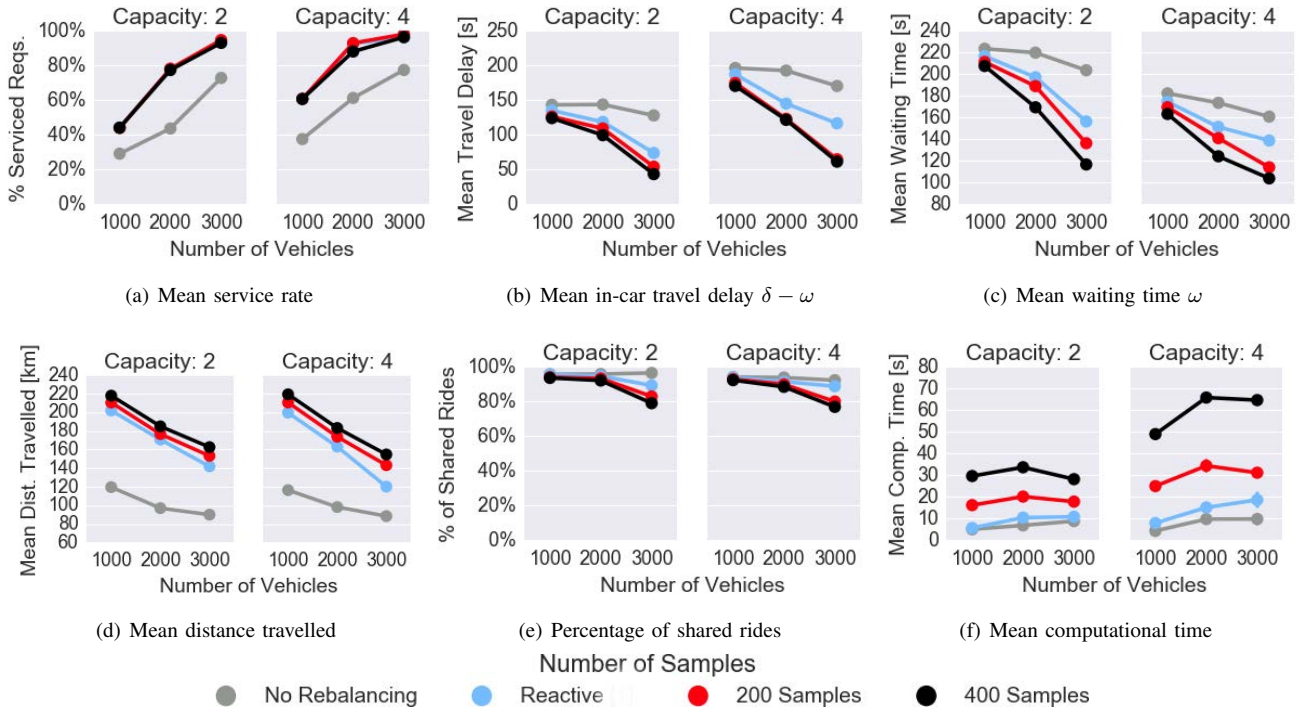
V. Evaluation

A. Experimental setup

We assess the performance of the system with a fleet of 1000, 2000, and 3000 vehicles of capacity two and four passengers. We used a fixed maximum waiting time of $\Omega = 5$ minutes and a maximum delay of $\Delta = 10$ minutes. The minimum inter-station distance used for the region discretization was 150 meters. For the experiments, we use one week of historical taxi trip data from 00:00 on Sunday May 5th, 2013 to 23:59 on Saturday May 11th, 2013 to assess the performance of our algorithm. This data comes from a publicly available source of all taxi trips in Manhattan, New York, USA [11]. This dataset contains the geographical coordinates for the origins and destinations along with the associated pick up and drop off dates and times for all trips in executed by the 13,586 active taxis in New York City. From this data we consider the request and pick up time to be equal since the time for the request is not publicly available.

In order to find routes for the taxis to execute, we consider the entire road network of Manhattan. We estimate the travel time for each road segment using the daily mean travel time computed by the method in [9]. Different travel times were used for weekdays, Saturday, and Sunday. The shortest paths using these travel times were precomputed between every two intersections in the road network and were stored in a look-up table.

We initialize the vehicles each day at midnight at sampled positions from the demand distribution. We then simulate the



(a) Mean service rate

(b) Mean in-car travel delay $\delta - \omega$

(c) Mean waiting time ω

(d) Mean distance travelled

(e) Percentage of shared rides

(f) Mean computational time

Fig. 4. Comparison of several performance metrics for varying number of sampled requests (No rebalancing, Reactive (0 samples), 200 samples, and 400 samples). The reactive method follows the algorithm of [10]. Each subplot corresponds to the vehicle capacity of 2 and 4 with the x-axis showing the fleet size (1000, 2000, and 3000 vehicles). We analyze (a) service rate (percentage of requests serviced), (b) average in car delay $\delta - \omega$, (c) average waiting time ω, (d) average distance travelled by each vehicle during a single day, (e) percentage of shared rides (number of passengers who shared a ride, divided by the total number of picked-up passengers) and (f) average computational time for a 30 seconds iteration of the method, in a 24 core 2.5GHz machine, including computation of the RV-graph, computation of the RTV-graph, ILP assignment including the sampled requests, rebalancing and writing the data to file.

execution of the fleet by issuing the requests obtained from the historical taxi dataset for the given day. The requests are collected within a 30 second time window after which they are assigned in batch to different vehicles using our algorithm of Sec. IV. In each time interval, or assignment step, we sample future requests up to 30 minutes in the future. We vary the number of predictions by using 0, 200, and 400 sampled predicted requests (per interval). These predicted requests enter the assignment problem of Algorithm 3, but are removed immediately afterwards, with new future requests being sampled in the following step. They do affect the assignment and routing at that time.

A pool of requests are kept until they have been picked up in case they can be reassigned to a better match. The number of requests in a single day varies from 382,779 on Sunday to 460,700 on Friday.

B. Results

We collect several metrics that characterize the system, including the service rate, in-car travel delay, waiting time, average distance traveled by the vehicles, percentage of shared rides, and the computational time. We use the same parameters as in [10], but with the additional sampled requests and cost term. These metrics are plotted for vehicle capacities two and four side by side in Fig. 4.

We observe that the service rate (number of requests serviced) remains approximately constant independently of the number of sampled requests, and it is close to 100% for 3,000 vehicles of capacity 4 (there are 13,000 active taxis per day in Manhattan). By sampling predicted requests we are able to reduce the mean in-car travel delay by 1.5 minutes and the mean waiting time by around 1 minute, with respect to the reactive approach.

Particularly, for the in-car travel delay and the waiting time, we see that there is a large benefit in using rebalancing and then a similar benefit by sampling predicted requests, see Fig.4-b) and -c). However, increasing the number of samples from 200 to 400 only marginally decreased the in-car travel delay by 3.4 seconds, when using a four passenger vehicle capacity and 3000 vehicles. It is likely that this small improvement is due to the time-outs introduced for real-time performance, which limit the benefit of additional samples. We believe that the increase would be larger if the algorithm was run to optimality.

We observe a trade-off between operational cost and performance, since the travel distance by the vehicles and the computational time of the approach do increase with the number of samples. The increase in travel distance arises from the fact that vehicles are routed towards predicted requests which may or may not appear in reality. This reduces mean waiting time and mean delay but does increase the miles traveled by each vehicle. The increase in computational time is due to the larger number of requests that enter the

routing and assignment problem. Furthermore, since they are in the future, they can be combined with many different trips, which leads to a potentially large number of feasible trips to be accounted for in the assignment. Nonetheless, the approach can be parallelized and would benefit from the large parallel servers available for fleet management companies.

To sum up, our experimental study confirms that the performance of a mobility-on-demand system with ride-sharing via our algorithm improves with knowledge of future demand. Yet, at a higher operational cost.

VI. Conclusion

In this paper we have presented a method for vehicle routing and request assignment inspired by [10] and which incorporates a prediction of future demand. The method seamlessly integrates sampled future requests in the request assignment and vehicle routing. We showed experimentally that the predictions do improve the positioning of the fleet of vehicles towards satisfying future requests, reducing waiting time and travel time. On the down side, we also observe that such an approach may increase the distance traveled by each vehicle and is computationally more expensive. Future works will aim at improving the rebalancing of vehicles, computing more accurate predictions including seasonal changes and online adaptation, and reduce the computational load. Besides intelligent transportation systems, we believe that this method also has great potential for other persistent multi-task multi-robot assignment problems.

References

[1] P. S. Castro, D. Zhang, and S. Li, "Urban traffic modelling and prediction using large scale taxi GPS traces," *International Conference on Pervasive Computing*, 2012.

[2] L. Moreira-Matias, J. Gama, and M. Ferreira, "On predicting the taxi-passenger demand: A real-time approach," *International Conference on Artificial Intelligence*, 2013.

[3] E. J. Gonzales, C. Yang, E. F. Morgul, and K. Ozbay, "Modeling Taxi Demand with GPS Data from Taxis and Transit," *Mineta National Transit Research Consortium*, 2014.

[4] 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.

[5] 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.

[6] K. Spieser, K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone, *Toward a systematic approach to the design and evaluation of automated mobility-on- demand systems: a case study in Singapore*. Road Vehicle Automation, 2014.

[7] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli, "Shared-vehicle mobility-on-demand systems: a fleet operator's guide to rebalancing empty vehicles," *Transportation Research Board 95th Annual Meeting, Washington, D.C.*, 2016.

[8] G. H. de Almeida Correia and B. van Arem, "Solving the user optimum privately owned automated vehicles assignment problem (uo-poavap): A model to explore the impacts of self-driving vehicles on urban mobility," *Transportation Research Part B: Methodological*, vol. 87, pp. 64–88, 2016.

[9] P. Santi, G. Resta, M. Szell, S. Sobolvesky, S. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences*, 2014.

[10] 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, pp. 462–467, Jan. 2017.

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

[12] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering Route Planning Algorithms," vol. 2, pp. 117–139, 2009.

[13] "2014 Yellow Taxi Trip Data." https://data.cityofnewyork.us/view/gn7m-em8n. Accessed: 2016-09-12.