On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment

Javier Alonso-Mora[a,1,2], Samitha Samaranayake[b], Alex Wallar[a], Emilio Frazzoli[c], and Daniela Rus[a]

[a]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139; [b]School of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853; and [c]Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139

Edited by Michael F. Goodchild, University of California, Santa Barbara, CA, and approved November 22, 2016 (received for review July 20, 2016)

Ride-sharing services are transforming urban mobility by providing timely and convenient transportation to anybody, anywhere, and anytime. These services present enormous potential for positive societal impacts with respect to pollution, energy consumption, congestion, etc. Current mathematical models, however, do not fully address the potential of ride-sharing. Recently, a large-scale study highlighted some of the benefits of car pooling but was limited to static routes with two riders per vehicle (optimally) or three (with heuristics). We present a more general mathematical model for real-time high-capacity ride-sharing that (*i*) scales to large numbers of passengers and trips and (*ii*) dynamically generates optimal routes with respect to online demand and vehicle locations. The algorithm starts from a greedy assignment and improves it through a constrained optimization, quickly returning solutions of good quality and converging to the optimal assignment over time. We quantify experimentally the trade-off between fleet size, capacity, waiting time, travel delay, and operational costs for low- to medium-capacity vehicles, such as taxis and van shuttles. The algorithm is validated with ∼3 million rides extracted from the New York City taxicab public dataset. Our experimental study considers ride-sharing with rider capacity of up to 10 simultaneous passengers per vehicle. The algorithm applies to fleets of autonomous vehicles and also incorporates rebalancing of idling vehicles to areas of high demand. This framework is general and can be used for many real-time multivehicle, multitask assignment problems.

ride-sharing | human mobility | vehicle routing | smart cities | intelligent transportation systems

New user-centric services are transforming urban mobility by providing timely and convenient transportation to anybody, anywhere, and anytime. These services have the potential for a tremendous positive impact on personal mobility, pollution, congestion, energy consumption, and thereby quality of life. The cost of congestion in the United States alone is roughly $121 billion per year or 1% of GDP (1), which includes 5.5 billion hours of time lost to sitting in traffic and an extra 2.9 billion gallons of fuel burned. These estimates do not even consider the cost of other potential negative externalities such as the vehicular emissions (greenhouse gas emissions and particulate matter) (2), travel-time uncertainty (3), and a higher propensity for accidents (4). Recently, the large-scale adoption of smart phones and the decrease in cellular communication costs has led to the emergence of a new mode of urban mobility, namely mobility-on-demand (MoD) systems, led by companies such as Uber, Lyft, and Via. These systems are able to provide users with a reliable mode of transportation that is catered to the individual and improves access to mobility to those who are unable to operate a personal vehicle, reducing the waiting times and stress associated with travel.

One of the major inefficiencies of current MoD systems is their capacity limitation, typically restricted to two passengers. Our method applies not only to shared taxis but also to shared vans and minibuses. A recent 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 few minutes (5). However, the method and analysis of ref. 5 was (*i*) limited to two riders for an optimal allocation (three with heuristics), (*ii*) intractable for larger number of passengers, and (*iii*) did not allow for allocation of additional riders after the start of a trip. There are no studies of this scale that quantify the benefits of larger-scale ride pooling, mainly due to the lack of efficient and scalable algorithms for this problem, both of which we address in this work.

Much of the fleet management literature for MoD systems considers the case of ride-sharing without pooling requests, focusing on fluid approximations (6), queuing based formulations (7), case studies in specific regions [e.g., Singapore (8)], and operational considerations for fleet managers (9). With the growing interest and rapid developments in autonomous vehicles, there is also an increasing focus on autonomous MoD systems (6, 9, 10). However, none of these works considered the ride-pooling problem of servicing multiple rides with a single trip. The ride-pooling problem is more related to the vehicle-routing problem and the dynamic pickup and delivery problem (11–15), where spatiotemporally distributed demand must be picked up and delivered within prespecified time windows. A major challenge when addressing this problem is the need to explore a very large decision space, while computing solutions fast enough to provide users with the experience of real-time booking and service.

Significance

Ride-sharing services can provide not only a very personalized mobility experience but also ensure efficiency and sustainability via large-scale ride pooling. Large-scale ride-sharing requires mathematical models and algorithms that can match large groups of riders to a fleet of shared vehicles in real time, a task not fully addressed by current solutions. We present a highly scalable anytime optimal algorithm and experimentally validate its performance using New York City taxi data and a shared vehicle fleet with passenger capacities of up to ten. Our results show that 2,000 vehicles (15% of the taxi fleet) of capacity 10 or 3,000 of capacity 4 can serve 98% of the demand within a mean waiting time of 2.8 min and mean trip delay of 3.5 min.

Author contributions: J.A.-M., S.S., and D.R. designed research; J.A.-M., S.S., E.F., and D.R. performed research; J.A.-M. and A.W. contributed new reagents/analytic tools; J.A.-M., S.S., A.W., E.F., and D.R. analyzed data; and J.A.-M., S.S., A.W., E.F., and D.R. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

Freely available online through the PNAS open access option.

[1]Present address: Delft Center for Systems and Control, Delft Technical University, 2628 CD, Delft, Netherlands.

[2]To whom correspondence should be addressed. Email: J.AlonsoMora@tudelft.nl.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1611675114/-/DCSupplemental.

Here, we consider the problem of using a fleet of vehicles with varying passenger capacities, and, in contrast to ref. 5, we address both the problems of assigning vehicles to matched passengers and rebalancing—or repositioning—the fleet to service demand. We show how the unified problem of passenger and vehicle assignment can be solved in a computationally efficient manner at a large scale, thereby demonstrating the capability to operate a real-time MoD system with multiple service tiers (shared-taxi, shared-vans, and shared-buses) of varying capacity.

Whereas previous approaches to this problem have focused on heuristic-based solutions (16–18), we present a reactive anytime optimal algorithm. That is, an algorithm that efficiently returns a valid assignment of travel requests to vehicles and then refines it over time, converging to an optimal solution. If enough computational resources are available, the optimal assignment for the current requests and time would be found; otherwise, the best solution found so far is returned.

Traditional approaches that rely on an integer linear program (ILP) formulation, such as ref. 19, also provide anytime guarantees for the multivehicle-routing problem. However, in contrast to our approach, their applicability is limited to small problem instances, which in ref. 19 was 32 requests and 4 vehicles, with a computation cost of several minutes. We also rely on an ILP formulation, but because we do not explicitly model the edges of the road network in the ILP, our approach scales to much larger problem instances. We observe that instances such as New York City, with thousands of vehicles, requests, and road segments, can be solved in real time.

Our approach decouples the problem by first computing feasible trips from a pairwise shareability graph (5) and then assigning trips to vehicles. We show that this assignment can be posed as an ILP of reduced dimensionality. The framework allows for flexibility in terms of prescribing constraints such as (but not limited to) maximum user waiting times and maximum additional delays due to sharing a ride. We also extend the method to proactively rebalance the vehicle fleet by moving idle vehicles to areas of high demand. In summary, we present a framework for solving the real-time ride-pooling problem with (*i*) arbitrary numbers of passengers and trips, (*ii*) anytime optimal rider allocation and routing dependent on the fleet location, and (*iii*) online rerouting and assignment of riders to existing trips.

We quantify experimentally the performance tradeoffs between fleet size, capacity, waiting time, travel delay, and operational costs for low- and medium-capacity vehicles (such as taxis, vans, or minibuses) in a large urban setting. Detailed experimental results are presented for a subset of ∼3 million rides extracted from the New York City taxicab public dataset. We show that 3,000 vehicles with a capacity of 2 and 4 could serve 94 and 98% of the demand with a mean waiting time of 3.2 and 2.7 min, and a mean delay of 1.5 and 2.3 min, respectively. To achieve 98% service rate, with comparable waiting time (2.8 min) and delay (3.5 min), a fleet of just 2,000 vehicles with a capacity of 10 was required. This fleet size is 15% of the active taxis in New York City (Movie S1). We also show that our approach is robust with respect to the density of requests and could therefore be applied to other cities.

Our system runs in real time and is particularly suited to autonomous vehicle fleets that can continuously reroute based on real-time requests. It can also rebalance idle vehicles to areas with high demand and is general enough to be applied to other multivehicle, multitask assignment problems.

Passenger Assignment and Vehicle Routing

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. We address the problems of both optimally assigning online travel requests to vehicles and finding optimal routes for the vehicle fleet. Each travel request consists of the time of request, a pickup location and a drop-off location.

We propose an anytime optimal algorithm for batch assignment of a set of requests $\mathcal{R}=\{r_1,\ldots,r_n\}$ to a set of vehicles $\mathcal{V}=\{v_1,\ldots,v_m\}$, which minimizes a cost function $\mathcal{C}$, satisfies a set of constraints $\mathcal{Z}$, and allows for multiple passengers per vehicle. A passenger is a past request that has been picked up by a vehicle and that is now en route to its destination. We denote by $\mathcal{P}_v$ the set of passengers for vehicle $v\in\mathcal{V}$. In a second step, the method also allows to rebalance the fleet of vehicles by driving idle vehicles to areas of high demand, where those vehicles are

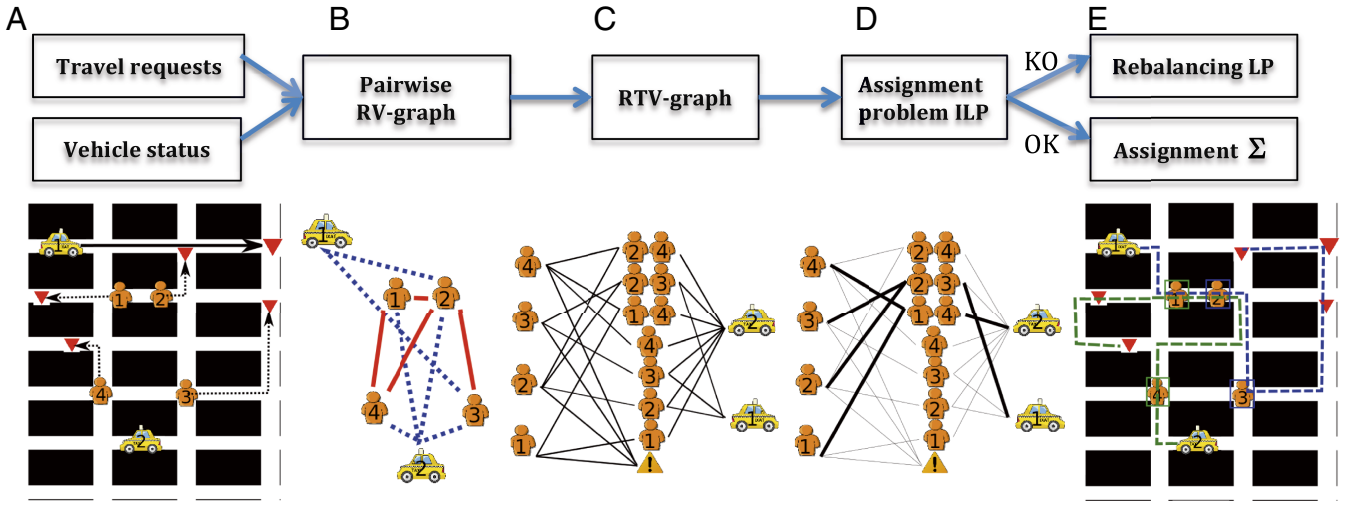

Fig. 1. Schematic overview of the proposed method for batch assignment of multiple requests to multiple vehicles of capacity ν. The method consists of several steps leading to an integer linear optimization that provides an anytime optimal assignment. (*A*) Example of a street network with four requests (orange human, origin; red triangle, destination) and two vehicles (yellow car, origin; red triangle, destination of 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 cannot be satisfied. (*D*) Optimal 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. In this case, no rebalancing step is required because all requests and vehicles are assigned.

likely to be required in the future. A schema of the method is shown in Fig. 1.

Our formulation is flexible with respect to physical and performance-related constraints that might need to be added. In our implementation, we consider the following. (*i*) For each request r, the waiting time ω_r, given by the difference between the pickup time t_r^p and the request time t_r^r, must be below a maximum waiting time Ω, for example, 2 min. (*ii*) For each passenger or request r the total travel delay $\delta_r = t_r^d - t_r^*$ must be lower than a maximum travel delay Δ, for example, 4 min, where t_r^d is the drop-off time and $t_r^* = t_r^r + \tau(o_r, d_r)$ is 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. Finally, (*iii*) for each vehicle v, we consider a maximum number of passengers, $n_v^{pass} \leq \nu$, for example, capacity 10.

We define the cost C of an assignment as the sum of delays δ_r (which includes the waiting time) over all assigned requests and passengers, plus a large constant c_{ko} for each unassigned request. Given an assignment Σ of requests to vehicles, we denote by $\mathcal{R}_{ok}$ the set of requests that have been assigned to some vehicle and $\mathcal{R}_{ko}$ the set of unassigned requests, due to the constraints or the fleet size. Formally,

$$C(\Sigma) = \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{P}_v} \delta_r + \sum_{r \in \mathcal{R}_{ok}} \delta_r + \sum_{r \in \mathcal{R}_{ko}} c_{ko}. \qquad [1]$$

This constrained optimization problem is solved via four steps (Fig. 1), which are: computing a pairwise request-vehicle shareability graph (RV-graph) (Fig. 1*B*); computing a graph of feasible trips and the vehicles that can serve them (RTV-graph) (Fig. 1*C*); solving an ILP to compute the best assignment of vehicles to trips (Fig. 1*D*); and rebalancing the remaining idle vehicles (Fig. 1*E*).

Given a network graph with travel times, we consider a function $travel(v, \mathcal{R}_v)$ for single-vehicle routing. For a vehicle v, with passengers $\mathcal{P}_v$, this function returns the optimal travel route σ_v to satisfy requests $\mathcal{R}_v$. This route minimizes the sum of delays $\sum_{r \in \mathcal{P}_v \cup \mathcal{R}_v} \delta_r$ subject to the constraints $\mathcal{Z}$ (waiting time, delay, and capacity). For low-capacity vehicles, such as taxis, the optimal path can be computed via an exhaustive search. For vehicles with larger capacity, heuristic methods such as Lin–Kernighan (20), Tabu search (21), or simulated annealing (22) may be used. Fig. 2, *Right* shows the optimal route for a vehicle with four passengers and an additional request.

The RV-graph (Fig. 1*B*) represents which requests and vehicles might be pairwise-shared and builds on the idea of shareability graphs proposed by ref. 5 but also includes the vehicles at their current state. Two requests r_1 and r_2 are connected if an empty virtual vehicle starting at the origin of one of them could pick up and drop off both requests while satisfying the constraints $\mathcal{Z}$. A cost $\delta_{r_1} + \delta_{r_2}$ is associated to each edge $e(r_1, r_2)$. Likewise, a request r and a vehicle v are connected if the request can be served by the vehicle while satisfying the constraints $\mathcal{Z}$, as given by $travel(v, r)$. The edge is denoted by $e(r, v)$.

Next, the cliques of the RV-graph—or regions for which its induced subgraph is complete—are explored to find feasible trips and compute the RTV-graph (Fig. 1*C*). A trip $T = \{r_1, \ldots, r_{n_T}\}$ is a set of n_T requests to be combined in one vehicle. A trip is feasible if all of the requests can be picked up and dropped off by some vehicle, while satisfying the constraints $\mathcal{Z}$.

This step computes feasible trips. There might be several trips of varying size that can service a particular request. In addition, more than one vehicle might be able to service a trip. The assignment step will later ensure that each request and vehicle are assigned to a maximum of one trip. The RTV-graph contains

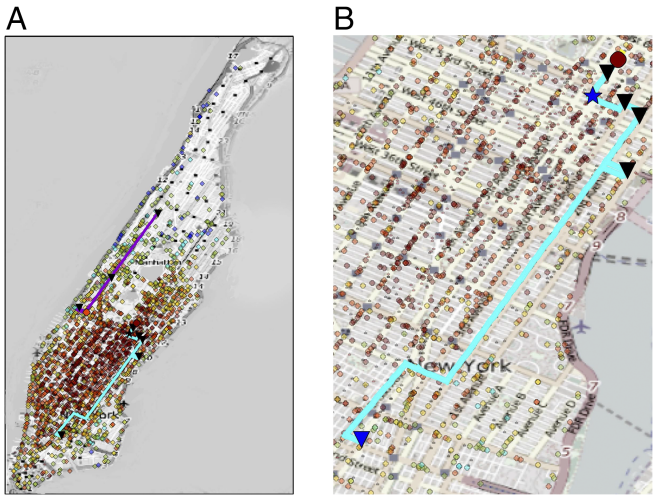

Fig. 2. (*A*) Snapshot: 2,000 vehicles, capacity of 4 ($\Omega = 5$ min, Wednesday, 2000 hours). Vehicle in the fleet are represented at their current positions. Colors indicate number of passengers (0: light blue; 1: light green; 2: yellow; 3: dark orange; 4: dark red); 39 rebalancing vehicles are displayed in dark blue—mostly in the upper Manhattan returning to the middle. (*B*) Close view of the scheduled path for a vehicle (dark red circle) with four passengers, which drops one off, picks up a new one (blue star), and drops all four. Drop-off locations are displayed with inverted triangles. See Movie S1 for a complete simulation.

two types of edges: (*i*) edges $e(r, T)$, between a request r and a trip T that contains request r (i.e., $\exists\ e(r, T) \Leftrightarrow r \in T$), and (*ii*) edges $e(T, v)$, between a trip T and a vehicle v that can execute the trip (i.e., $\exists\ e(T, v) \Leftrightarrow travel(v, T)$ is feasible). The cost $\sum_{r \in \mathcal{P}_v \cup T} \delta_r$, sum of delays, is associated to each edge $e(T,v)$.

The algorithm to compute the feasible trips and edges proceeds incrementally in trip size for each vehicle, starting from the request-vehicle edges in the RV-graph (*SI Appendix*, Algorithm 1). For computational efficiency, we rely on the fact that a trip T only needs to be checked for feasibility if there exists a vehicle v for which all of its subtrips $T' = T \setminus r$ (obtained by removing one request) are feasible and have been added as edges $e(T', v)$ to the RTV-graph.

Next, we compute the optimal assignment Σ_{optim} of vehicles to trips. This optimization is formalized as an ILP, initialized with a greedy assignment obtained directly from the RTV-graph. To compute the greedy assignment Σ_{greedy}, trips are assigned to vehicles iteratively in decreasing size of the trip and increasing cost (sum of travel delays). The idea is the maximize the amount of requests served while minimizing the cost (*SI Appendix*, Algorithm 2).

The optimization problem is formulated in Algorithm 1. A binary variable $\epsilon_{i,j} \in \{0, 1\}$ is introduced for each edge $e(T_i, v_j)$ between a trip $T_i \in \mathcal{T}$ and a vehicle $v_j \in \mathcal{V}$ in the RTV-graph. If $\epsilon_{i,j} = 1$, then vehicle v_j is assigned to trip T_i. We denote by $\mathcal{E}_{TV}$ the set of $\{i, j\}$ indices for which an edge $e(T_i, v_j)$ exists in the RTV-graph, i.e., the set of possible pickup trips. An additional binary variable $\chi_k \in \{0, 1\}$ is introduced for each request $r_k \in \mathcal{R}$. These variables are active, i.e., $\chi_k = 1$, if the associated request r_k can not be served by any vehicle and is ignored. The set of variables is then $\mathcal{X} = \{\epsilon_{i,j}, \chi_k; \forall\ e(T_i, v_j)$ edge in RTV-graph and $\forall\ r_k \in \mathcal{R}\}$.

The cost terms $c_{i,j}$ are the sum of delays for trip T_i and vehicle v_j pickup (stored in the $e(T_i, v_j)$ edge of the RTV-graph) and c_{ko} is a large constant to penalize ignored requests.

Two types of constraints are included. Line 3 in Algorithm 1 imposes that each vehicle is assigned to one trip at most. Line 4 in Algorithm 1 imposes that each request is assigned to a single vehicle or ignored. In these constraints, three sets appear. The set of trips that can be serviced by a vehicle j, or edges $e(T_i, v_j)$, is $\mathcal{I}^T_{V=j}$. The set of trips that contain request k, or edges $e(r_k, T_i)$, is $\mathcal{I}^T_{R=k}$. The set of vehicles that can service trip i, or edges $e(T_i, v_j)$, is $\mathcal{I}^V_{T=i}$.

This ILP is solved incrementally from the greedy assignment Σ_{greedy}, improving the quality of the assignment over time.

Algorithm 1. Optimal assignment

1: Initial guess: Σ_{greedy}

2: $\Sigma_{optim} := \arg\min_{\mathcal{X}} \sum_{i,j \in \mathcal{E}_{TV}} c_{i,j}\epsilon_{i,j} + \sum_{k \in \{1,\dots,n\}} c_{ko}\chi_k$

3: s.t. $\sum_{i \in \mathcal{I}^T_{V=j}} \epsilon_{i,j} \leq 1 \qquad \forall\, v_j \in \mathcal{V}$

4: $\sum_{i \in \mathcal{I}^T_{R=k}} \sum_{j \in \mathcal{I}^V_{T=i}} \epsilon_{i,j} + \chi_k = 1 \qquad \forall\, r_k \in \mathcal{R}$

This method is well suited for online execution to assign incoming requests $r(t)$ to a fleet of vehicles for which a pool of requests $\mathcal{R}$ is maintained where (*i*) new requests are added as they are received and (*ii*) requests are removed when they are either (*a*) picked up by a vehicle or (*b*) could not be successfully matched to any vehicle within the maximum waiting time (they are ignored).

Requests are collected during a time window (e.g., 30 s), after which they are assigned in batch to the different vehicles. If a request is matched to a vehicle at any given iteration, its latest pickup time is reduced to the expected pickup time by that vehicle and the cost χ_{ko} of ignoring it is increased for subsequent iterations. A request might be rematched to a different vehicle in subsequent iterations as long as its waiting time does not increase and until it is picked up by some vehicle. Once a request is picked up, it remains in that vehicle and cannot be rematched—the vehicle may still pick additional passengers. In each iteration, the new assignment of requests to vehicles guarantees that the current passengers are dropped off within the maximum delay constraint.

After the assignment, due to fleet imbalances, the set $\mathcal{R}_{ko}$ of unassigned requests may not be empty, and some empty vehicles $\mathcal{V}_{idle}$ may still by unassigned to any request. These imbalances may occur when the idle vehicles are in areas far away from the area of current requests and due to the maximum waiting time and delay constraints and vehicle capacity. Under the assumptions that (*i*) ignored requests may wait longer and request again, (*ii*) it is likely that more requests occur in the same area where all requests cannot be satisfied, and (*iii*) there are not enough requests in the neighborhood of the idle cars, we propose the following approach to rebalance the fleet by moving only the idle vehicles.

To rebalance the vehicle fleet, after each batch assignment, the vehicles in $\mathcal{V}_{idle}$ are assigned to requests in $\mathcal{R}_{ko}$ to minimize the sum of travel times, with the constraint that either all requests or all of the vehicles are assigned. We first compute the travel time of each individual idle vehicle in $\mathcal{V}_{idle}$ to pick each ignored request in $\mathcal{R}_{ko}$ and then obtain the optimal assignment via a linear program (*SI Appendix*, Algorithm 4). In this approach, if all requests can be satisfied, some vehicles may remain idle, saving fuel and distance traveled, which is the case at nighttime.

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 the worst case, the number of variables is of order $\mathcal{O}(mn^\nu)$ but 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. The number of constraints is $n + m$.

Anytime Optimality. This method guarantees optimality of the assignment of the currently active requests, while satisfying the constraints $\mathcal{Z}$, if all of the steps are executed until termination and exploration of all possible trips and assignments. In practice, timeouts can be set both for the amount of time spent generating candidate trips for each vehicle and for the 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 optimality for tractability, and their values will depend on the available resources. We note that the method is reactive, in the sense that it provides anytime-optimality guarantees given the current state of the system and the current requests. To inform the assignment and routing about future demand, an additional cost term could be added to Eq. **1**, and future requests could be sampled from historical data. The method allows for parallelization in all steps. Proofs are provided in *SI Appendix*, *III. Theoretical Guarantees*.

Results

We assess the performance of a MoD fleet controller using the proposed algorithm, against real data from an arbitrarily chosen representative week, from 0000 hours Sunday, May 5, 2013, to 2359 hours, Saturday May 11, 2013, from the publicly available dataset of taxi trips in Manhattan, New York City (23). This dataset contains for each day the time and location of all of the pickups and drop-offs executed by each of the 13,586 active taxis. From these data, we extract all of the requests (origin and destination within Manhattan) and consider the time of request equal to the time of pickup. We consider the complete road network of Manhattan (4,092 nodes and 9,453 edges), with the travel time on each edge (road segment) of the network given by the daily mean travel time estimate, computed using the method in ref. 5. Shortest paths and travel times between all nodes are then precomputed and stored in a lookup table.

We perform a simulation of the evolution of the taxi fleet, where vehicles are initialized at midnight at sampled positions from a historical demand distribution and continuously travel to pick up and drop off passengers to satisfy the real requests extracted from the dataset. Requests are collected during a

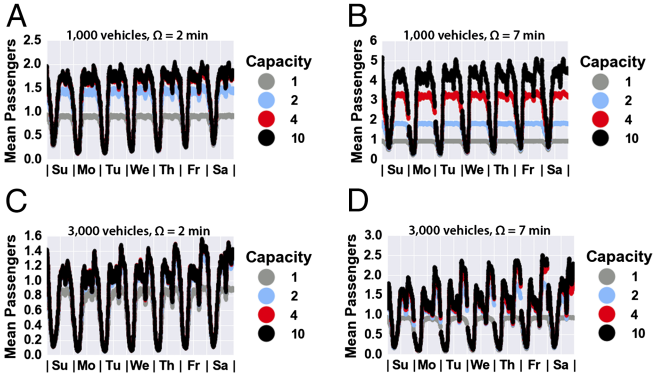

Fig. 3. Mean number of passengers per vehicle for four different vehicle types (capacity one, two, four, and ten). We show four one-week time series for different fleet sizes and maximum waiting time: (*A*) 1000 vehicles and $\Omega = 2$ min; (*B*) 1000 vehicles and $\Omega = 7$ min; (*C*) 3000 vehicles and $\Omega = 2$ min; and (*D*) 3000 vehicles and $\Omega = 7$ min. At night, most vehicles wait, and during rush hour, the mean occupancy decreases as the fleet gets larger. Larger maximum waiting time enables more opportunities for ride-sharing.

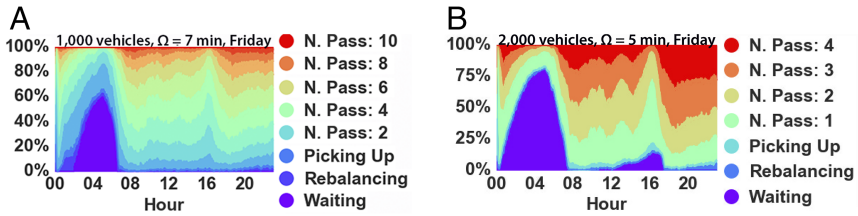

Fig. 4. Percentage of vehicles in each state (waiting, rebalancing, and number of passengers) for a representative day (Friday 0000 hours to 2400 hours). (*A*) A fleet of 1,000 vehicles of capacity 10 with many opportunities for ride-sharing in high-capacity vehicles. (*B*) A fleet of 2,000 vehicles of capacity four, showing the utility of full vehicle-sharing. Additional figures, for varying days and parameters, are in *SI Appendix, VIII. Additional Experimental Figures*.

time window, 30 s in our experiments, after which they are assigned in batch to the different vehicles. Past requests are kept in the requests pool until picked up and can be reassigned if a better match is found before pickup. Each day contains between 382,779 (Sunday) and 460,700 (Friday) requests, and the running pool of requests contains up to 2,000 requests at any given time. The method is robust both with respect to the chosen time window and the density of demands, as shown in *SI Appendix, VI. Robustness Analysis* in results with a time window between 10 and 50 s, and with half/double the amount of requests (∼220,000/∼880,000 per day) in New York City.

We analyze several metrics, with different vehicle fleet sizes ($m \in \{1{,}000, 2{,}000, 3{,}000\}$ vehicles), vehicle capacities ($\chi \in \{1, 2, 4, 10\}$ passengers), and maximum waiting times ($\Omega \in \{120, 300, 420\}$ s). The maximum trip delay Δ is double the maximum waiting time and includes both the waiting time ω and the inside-the-vehicle travel delay. Our analysis shows that, thanks to high-capacity ride-sharing, a reduced fleet of vehicles (below 25% of the active taxis in New York City) is able to satisfy 99% of the requests, with a mean waiting time and delay of about 2.5 min. All results in this section include rebalancing of idle vehicles to unassigned requests; experimentally, we observed that the rebalancing step contributed an increase in the service rate of about 20% (*SI Appendix*, Table II). Movie S1 shows the evolution of the taxi fleet in New York City for a subset of experiments.

High vehicle occupancy is achieved in times of high demand, with a large number of the trips being shared. In Fig. 2, we observe that many vehicles are located in mid-Manhattan and contain three or four passengers. Fig. 3 shows that the occupancy depends on the fleet size, capacity, and the maximum waiting/delay time. Lower fleet size, larger capacity and longer waiting/delay times increase the possibilities for ride-sharing and lead to higher mean vehicle occupancy. In Fig. 4, we observe that during peak hours, a small fleet of high-capacity vehicles does indeed operate at high occupancy. For a fleet of 1,000 vehicles of capacity 10, we observe that, during peak time (1800 hours) of a Friday, 10% of the vehicles have eight or more passengers, 40% of the vehicles have six or more, 80% have three or more, and 98% have at least one passenger. For a fleet of 2,000 vehicles of capacity four, we observe that, at the same peak time, over 70% of them have at least three passengers onboard.

We observe that the value of fleets with larger passenger capacities increases with larger Ω and Δ values, as expected, because passengers are willing to incur a larger personal time penalty. High-capacity vehicles are also more important when the fleet size is smaller, because seating capacity might be a bottleneck with smaller fleets. For instance (Fig. 5*A*), a fleet of 1,000 vehicles with a capacity of 10 can satisfy almost 80% of the requests with $\Omega = 420$ s, compared with below 30% for a single-rider taxi, for a net gain of over 50%. However, with a larger fleet of 3,000 vehicles and $\Omega = 120$ s, the benefit is only about 15%. Interestingly, if longer waiting times and delays are

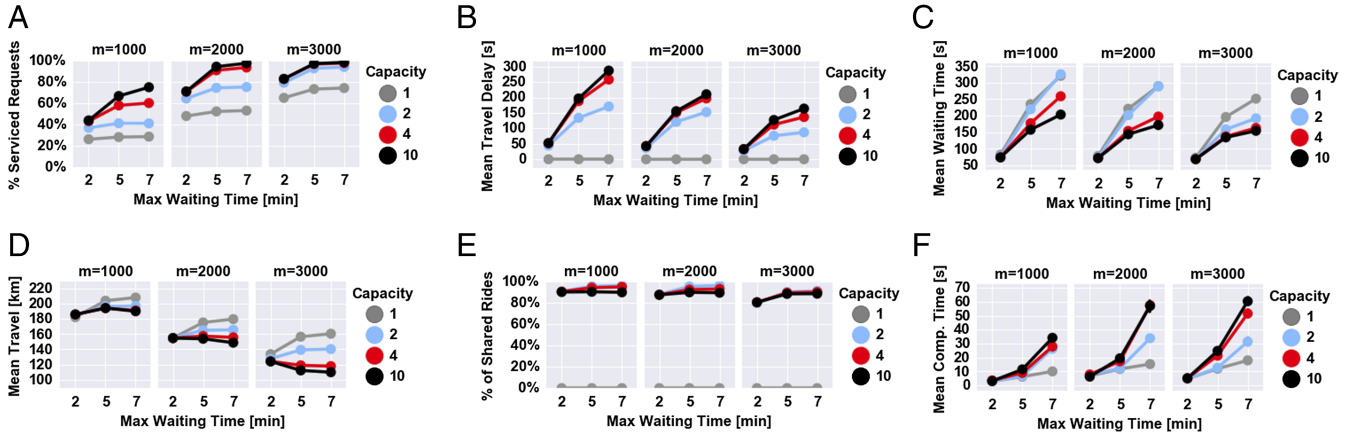

Fig. 5. Comparison of several performance metrics for varying vehicle capacity (1, 2, 4, and 10 passenger, shown with lines). Each subplot is for a fleet size of 1,000, 2,000, and 3,000 vehicles, and the coordinate axes show increasing maximum waiting time Ω of 2, 5, and 7 min. We analyze service rate (percentage of requests serviced) (*A*), average in car delay $\delta - \omega$ (*B*), average waiting time ω (*C*), average distance traveled by each vehicle during a single day (*D*), percentage of shared rides (number of passengers who shared a ride divided by the total number of picked-up passengers) (*E*), and average computational time for a 30-s iteration of the method (*F*), in a 24 core 2.5 GHz machine, including computation of the RV-graph, computation of the RTV-graph, ILP assignment, rebalancing, and data writing (higher levels of parallelization would drastically reduce this computational time). The parameters used in the simulation are specified in *SI Appendix, III. Theoretical Guarantees, C. Heuristics for Real-Time Execution*.

allowed, $\Omega = 420$ s, a fleet of 3,000 vehicles with a capacity of 2, 4, and 10 could serve 94, 98, and 99% of the demand. To achieve 98% service rate, a fleet of just 2,000 vehicles with a capacity of 10 was required, which represents a reduction of the fleet size to 15% of the active taxi fleet in New York City.

As expected, the in-car travel delay does increase with the increase in vehicle capacity (Fig. 5*B*). Nonetheless, that increase seems practically negligible—well below 100 s—once ride-sharing is allowed. Furthermore, the mean waiting time does in fact decrease as vehicle capacity is increased (Fig. 5*C*). For a fleet size of 1,000 vehicles and $\Delta = 420$ s, high-capacity vehicles not only improved the service rate but also achieved a reduction in mean waiting time of over 100 s, which partially offsets the increased in-car delay. In particular, we observe that 3,000 vehicles with a capacity of 2 and 4 could serve 94 and 98% of the demand, with a mean waiting time of 3.2 and 2.7 min and a mean delay of 1.5 and 2.3 min, respectively. To achieve 98% service rate, with comparable waiting time (2.8 min) and delay (3.5 min), a fleet of just 2,000 vehicles with a capacity of 10 was required.

We also observed that increasing the vehicle capacity not only increases the service rate but also reduces the mean distance traveled by the vehicles in the fleet (Fig. 5*D*), potentially leading to a reduction in costs, congestion, and pollution. We also observe that, with our online method, about 90% of the rides were shared. The number of shared rides slightly increases with Δ and decreases with the fleet size (Fig. 5*E*). Finally, we note that our approach is real-time capable (Fig. 5*F*). In our setup, for $\Omega \leq 300$ s, the method is executed in less that 30 s, which is the period for which requests are collected.

Conclusion

In this paper, we introduced a reactive anytime optimal method with scalable real-time performance for assigning passenger requests to a fleet of vehicles of varying capacity. We quantify experimentally the tradeoff between fleet size, capacity, waiting time, travel delay, and operational costs for low- and medium-capacity vehicles, such as taxis or vans in a large-scale city dataset. Under the assumption of one person per ride, we show that 98% of the taxi rides currently served by over 13,000 taxis could be served with just 3,000 taxis of capacity four. We observe that a vehicle capacity of two is sufficient for ride-sharing when a small trip delay of 2 min is imposed. If a maximum delay of 5 min or more (comparable to the time spent retrieving a car from parking) is allowed, higher-capacity vehicles (*i*) increase the service rate significantly, (*ii*) reduce the waiting time, and (*iii*) reduce the distance traveled by each vehicle. Our analysis shows that a ride-pooling service can provide a substantial improvement in urban transportation systems and that the system parameters such as vehicle capacity and fleet size depend on quality of service requirements and demand.

ACKNOWLEDGMENTS. We thank G. Resta, P. Santi, and C. Ratti for sharing the graph of Manhattan and the estimated travel times of ref. 5. This work was supported in part by the Office of Naval Research Grant N00014-12-1-1000 and the Massachusetts Institute of Technology–Singapore Alliance on Research and Technology under the Future of Urban Mobility.

1. Schrank D, Eisele B, Lomax T (2012) Texas Transportation Institute 2012 Urban Mobility Report (Texas Transportation Institute, A&M University, College Station, TX).
2. Pant P, Harrison RM (2013) Estimation of the contribution of road traffic emissions to particulate matter concentrations from field measurements: a review. *Atmos Environ* 77:78–97.
3. Carrion C, Levinson D (2012) Value of travel time reliability: a review of current evidence. *Transp Res Part A Policy Pract* 46(4):720–741.
4. Hennessy DA, Wiesenthal DL (1999) Traffic congestion, driver stress, and driver aggression. *Aggress Behav* 25(6):409–423.
5. Santi P, et al. (2014) Quantifying the benefits of vehicle pooling with shareability networks. *Proc Natl Acad Sci USA* 111(37):13290–13294.
6. Pavone M, Smith SL, Frazzoli E, Rus D (2012) Robotic load balancing for mobility-on-demand systems. *Int J Rob Res* 31(7):839–854.
7. Zhang R, Pavone M (2014) Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *Proceedings of Robotics: Science and Systems Conference*, July 12–16, 2014, Berkeley, CA.
8. Spieser K, et al. (2014) Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in Singapore. *Road Vehicle Automation* (Springer International Publishing, Cham, Switzerland), pp 229–245.
9. Spieser K, Samaranayake S, Gruel W, Frazzoli E (2016) Shared-vehicle mobility-on-demand systems: A fleet operator's guide to rebalancing empty vehicles. Transportation Research Board 95th Annual Meeting, January 10–14, 2016, Washington, DC, abstr 16-5987.
10. de Almeida Correia GH, van Arem B (2016) 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. *Transp Res Part B Methodological* 87:64–88.
11. Toth P, Vigo D (2014) Vehicle Routing: Problems, Methods, and Applications (SIAM, Philadelphia) Vol 18.
12. Pillac V, Gendreau M, Guéret C, Medaglia AL (2013) A review of dynamic vehicle routing problems. *Eur J Oper Res* 225(1):1–11.
13. Berbeglia G, Cordeau JF, Laporte G (2010) Dynamic pickup and delivery problems. *Eur J Oper Res* 202(1):8–15.
14. Golden BL, Raghavan S, Wasil EA (2008) The Vehicle Routing Problem: Latest Advances and New Challenges (Springer Science & Business Media, Berlin) Vol 43.
15. Stenger A, Vigo D, Enz S, Schwind M (2013) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transport Sci* 47(1):64–80.
16. Agatz NAH, Erera AL, Savelsbergh MWP, Wang X (2011) Dynamic ride-sharing: a simulation study in metro Atlanta. *Transp Res Part B Meth* 45(9):1450–1464.
17. Horn MET (2002) Fleet scheduling and dispatching for demand-responsive passenger services. *Transp Res Part C Emerg Technol* 10(1):35–63.
18. Ma S, Zheng Y, Wolfson O (2013) T-share: A large scale dynamic taxi ridesharing service. *2013 IEEE 29th International Conference on Data Engineering (ICDE)* (IEEE, Piscataway, NJ), pp 410–421.
19. Cordeau JF (2006) A branch-and-cut algorithm for the dial-a-ride problem. *Oper Res* 54(3):573–586.
20. Helsgaun K (2000) An effective implementation of the Lin–Kernighan traveling salesman heuristic. *Eur J Oper Res* 126(1):106–130.
21. Glover F, Laguna M (2013) Tabu Search? (Springer, Berlin).
22. Pham DT, Karaboga D (2012) Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks (Springer Science & Business Media, Berlin).
23. Donovan B, Work DB (2015) Using coarse GPS data to quantify city-scale transportation system resilience to extreme events. arXiv:1507.06011.