

## Large-scale online ridesharing the effect of assignment optimality on system performance

Fiedler, David; Čertický, Michal; Alonso-Mora, Javier; Pěchouček, Michal; Čáp, Michal

### DOI

[10.1080/15472450.2022.2121651](https://doi.org/10.1080/15472450.2022.2121651)

### Publication date

2022

### Document Version

Final published version

### Published in

Journal of Intelligent Transportation Systems: technology, planning, and operations

### Citation (APA)

Fiedler, D., Čertický, M., Alonso-Mora, J., Pěchouček, M., & Čáp, M. (2022). Large-scale online ridesharing: the effect of assignment optimality on system performance. *Journal of Intelligent Transportation Systems: technology, planning, and operations*, 28 (2024)(2), 189-210.  
<https://doi.org/10.1080/15472450.2022.2121651>

### Important note

To cite this publication, please use the final published version (if applicable).  
Please check the document version above.

### Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

### Takedown policy

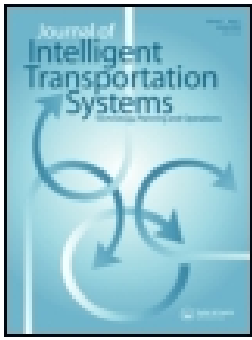
Please contact us and provide details if you believe this document breaches copyrights.  
We will remove access to the work immediately and investigate your claim.

***Green Open Access added to TU Delft Institutional Repository***

***'You share, we take care!' - Taverne project***

**<https://www.openaccess.nl/en/you-share-we-take-care>**

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.



# Large-scale online ridesharing: the effect of assignment optimality on system performance

David Fiedler, Michal Čertický, Javier Alonso-Mora, Michal Pěchouček & Michal Čáp

To cite this article: David Fiedler, Michal Čertický, Javier Alonso-Mora, Michal Pěchouček & Michal Čáp (2022): Large-scale online ridesharing: the effect of assignment optimality on system performance, Journal of Intelligent Transportation Systems, DOI: [10.1080/15472450.2022.2121651](https://doi.org/10.1080/15472450.2022.2121651)

To link to this article: <https://doi.org/10.1080/15472450.2022.2121651>



Published online: 04 Dec 2022.



Submit your article to this journal [↗](#)






View related articles [↗](#)



View Crossmark data [↗](#)



# Large-scale online ridesharing: the effect of assignment optimality on system performance

David Fiedler<sup>a</sup> , Michal Čertický<sup>a</sup>, Javier Alonso-Mora<sup>b</sup> , Michal Pěchouček<sup>a</sup> , and Michal Čáp<sup>a</sup>

<sup>a</sup>Department of Computer Science, Faculty of Electrical Engineering, CTU in Prague, Prague, Czech Republic; <sup>b</sup>Department of Cognitive Robotics, TU Delft, Delft, The Netherlands

## ABSTRACT

Mobility-on-demand (MoD) systems consist of a fleet of shared vehicles that can be hailed for one-way point-to-point trips. The total distance driven by the vehicles and the fleet size can be reduced by employing ridesharing, i.e., by assigning multiple passengers to one vehicle. However, finding the optimal passenger-vehicle assignment in an MoD system is a hard combinatorial problem. In this work, we demonstrate how the VGA method, a recently proposed systematic method for ridesharing, can be used to compute the optimal passenger-vehicle assignments and corresponding vehicle routes in a massive-scale MoD system. In contrast to existing works, we solve all passenger-vehicle assignment problems to *optimality*, regularly dealing with instances containing thousands of vehicles and passengers. Moreover, to examine the impact of using optimal ridesharing assignments, we compare the performance of an MoD system that uses optimal assignments against an MoD system that uses assignments computed using insertion heuristic and against an MoD system that uses no ridesharing. We found that the system that uses optimal ridesharing assignments subject to the maximum travel delay of 4 minutes reduces the vehicle distance driven by 57% compared to an MoD system without ridesharing. Furthermore, we found that the optimal assignments result in a 20% reduction in vehicle distance driven and 5% lower average passenger travel delay compared to a system that uses insertion heuristic.

## ARTICLE HISTORY

Received 9 December 2020  
Revised 5 July 2022  
Accepted 10 July 2022

## KEYWORDS

Mobility-on-demand; ride-sharing; simulation; traffic control; vehicle routing

## 1. Introduction

In densely populated cities, private cars are considered as an unsustainable mode of transportation. Typically, parking capacity and road capacity are insufficient to accommodate all private transport and, at the same time, difficult to expand due to lack of available urban space or high cost. As a result, many modern cities suffer from traffic congestion, unavailability of parking spaces, and air pollution.

One of the proposed solutions to address these problems is the deployment of metropolitan mobility-on-demand (MoD) systems providing an alternative to traveling in a private vehicle designed to be as comfortable as traveling in a private car but with smaller parking capacity and road capacity requirements (Alonso-Mora et al., 2017; Čáp & Alonso-Mora, 2018; Miller & How, 2017; Spieser et al., 2014). These MoD systems consist of a fleet of shared passenger vehicles that jointly serve the travel requests of the system's users. Usually, these systems are assumed to use traditional vehicles able to carry up to four

passengers simultaneously. For each incoming travel request, the MoD system assigns the request to one of the vehicles and alters its route such that the passenger is picked up and transported to the drop-off location. Examples of such an MoD system include services like Uber Pool or Lyft Line, as well as the future systems of autonomous self-driving cars being developed by companies such as Waymo, Cruise, or Motional.

Such MoD systems employ *vehicle sharing*, so they can serve the existing transportation demand with a smaller, highly-utilized vehicle fleet and thus, significantly reduce the need for urban parking space. To further improve the system's efficiency, the provider can implement *ridesharing*, where multiple passengers can be transported in one vehicle simultaneously (Alonso-Mora et al., 2017). Efficient ridesharing increases vehicle occupancy, which consequently reduces the required fleet size and total distance driven by the vehicle fleet, resulting in ecological and economic benefits.

Two clarifying notes on terminology are in order. First, we note that MoD systems are ultimately envisioned also to include high-capacity transportation modes (e.g., trains, subway, buses) and to allow for transfers between different vehicles (Shaheen & Cohen, 2020). However, this article focuses on MoD systems that transport each passenger from their pick-up to their destination in one vehicle. Analysis of MoD systems that allow transfers is left for future work. Second, we also caution that the term *ridesharing* is overloaded. This paper focuses on *ridesharing* in on-demand mobility systems, where each vehicle is driven by a for-hire driver who transports travelers between their desired pick-up and drop-off locations. Alternatively, in the future, these vehicles may be self-driving. Apart from that, there is a distinct concept called *peer-to-peer ridesharing*, where the vehicle is typically owned and driven by one of the travelers, whose primary motivation is to reach his/her intended destination. Readers interested in peer-to-peer ridesharing are referred to the growing body of research devoted to this model, for example, the work of Li et al. (2019) that studies the impact of high occupancy toll lane configurations on the willingness of (peer) drivers to share a ride, or J. Ma et al. (2020) and Yan et al. (2019) who study the ridesharing user equilibrium in the context of peer-to-peer ridesharing.

### 1.1. Related work

Recently, a number of mobility-on-demand system models have been developed with the aim to provide quantitative insights into the potential of large-scale carsharing and ridesharing to improve the efficiency of urban transportation.

Most existing models of MoD systems assume unit-capacity vehicles (Bischoff & Maciejewski, 2016; Fiedler et al., 2017; Maciejewski & Bischoff, 2018; Spieser et al., 2014; Venkatraman & Levin, 2019). However, transportation systems that do not employ ridesharing suffer from poor operational efficiency because the vehicles need to travel empty from the drop-off point of a passenger to the pick-up point of the following passenger. Such unallocated trips can generate significant extra vehicular traffic in the system; various studies indicate the growth in vehicle distance traveled from 17% to 40% depending on the system configuration (Bischoff & Maciejewski, 2016; Fiedler et al., 2017; Maciejewski & Bischoff, 2018). The average vehicle occupancy observed in such systems is considerably lower than one passenger per vehicle (Fiedler et al., 2018), a finding which also

corresponds to the average vehicle occupancy measured in already operating taxi services (NYC Taxi & Limousine Commission, 2016). The low occupancy in MoD systems can lead to congestion, which could be partially alleviated by a congestion-aware dispatching (Venkatraman & Levin, 2019).

Therefore, it is beneficial to consider vehicles with a capacity higher than one and allow ridesharing between passengers. In contrast to peer-to-peer ridesharing (Li et al., 2019; Masoud & Jayakrishnan, 2017; Tamannaie & Irandoost, 2019), here we are interested in the centralized setting, where a central dispatcher decides on an efficient assignment of travel requests to fleet vehicles. This problem is commonly formulated as a Vehicle Routing Problem with Pickup and Deliveries (VRPPD) or, more specifically, as Dial-a-Ride Problem (DARP) (Cordeau & Laporte, 2007; Toth & Vigo, 2014). These formulations can be solved optimally using off-the-shelf Integer Linear Programming (ILP) solvers or domain-tailored ILP solution techniques. However, the applicability of these methods is limited to small-scale instances with at most tens of requests and vehicles. Large-scale MoD systems typically require the ability to find routes for many more vehicles and requests. For example, in New York City (NYC), there are almost 100,000 active taxis per hour during peak traffic (NYC Taxi & Limousine Commission, 2018). Therefore, DARP instances appearing in large-scale MoD systems are typically solved using heuristic methods.

A popular heuristic method for large-scale dynamic DARP is the Insertion Heuristic (IH) (Bischoff et al., 2017; Campbell & Savelsbergh, 2004; Fiedler et al., 2018; Kalina et al., 2015). Also, IH is often used as a subcomponent of more sophisticated algorithms. For example, in ridesharing with demand prediction (van Engelen et al., 2018), when integrating ridesharing with public transport (T.-Y. Ma et al., 2019), or as an initial solution generator for metaheuristic methods (Muelas et al., 2013). The metaheuristic methods, which are effective in solving conventional DARP problem instances (Ho et al., 2018), typically target scenarios with less than twenty vehicles (Masmoudi et al., 2016; Pfeiffer & Schulz, 2022) and suffer from scalability issues when applied to large-scale DARPs. However, in the last decade, there was some progress with metaheuristic approaches enabling solving larger instances. Jung et al. (2016) used simulated annealing to solve scenarios with 600 operating vehicles. Another popular metaheuristic is the Greedy Randomized Adaptive Search Procedure (GRASP)

used by Santos and Xavier (2013). The authors were able to solve instances with up to 750 requests. They also tested an online setting with 78,000 requests per day, and later, they improved the results significantly (Santos & Xavier, 2015). Muelas et al. (2013) also solved four types of specialized DARP scenarios with up to 90 vehicles using Variable Neighborhood Search. Later, Muelas et al. (2015) modified this approach to a distributed version which was able to solve scenarios with up to 1668 vehicles and 16,000 requests. Another metaheuristic, a modified artificial bee colony algorithm, was used by Zhan et al. (2021). The method was able to solve an instance of 3661 requests and 2400 vehicles. Later, this method was used in a simulation-optimization framework for an MoD system with electric vehicles (Zhan et al., 2022).

A systematic and scalable approach for pairwise ridesharing based on bipartite matching in the so-called shareability network was proposed by Santi et al. (2014). The analysis revealed that up to 80% of the trips could be pairwise shared while keeping the travel delay lower than a couple of minutes. Later, Alonso-Mora et al. (2017) proposed a new method that lifted the limit of two passengers per car and evaluated this method on the NYC taxi dataset. Finally, Čáp and Alonso-Mora (2018) utilized this method to study the tradeoffs between the quality of service and the operation cost inherent in ridesharing.

Finally, apart from optimizing the assignment of passengers to vehicles, we can also optimize the pickup and drop-off locations if we enable short walking for passengers. Such an approach was tested by Fielbaum et al. (2021), showing that it can improve the level of service and decrease the total travel time. Later, Fielbaum (2021) tried optimizing pickup and drop-off positions of precomputed vehicle plans to measure the benefits exactly. He demonstrated that we could decrease the travel cost by almost 19% when optimizing the locations with a heuristic method and 24% with a slower optimal solution method.

## 1.2. Contribution

In this work, we extend the existing study of ridesharing in large-scale MoD systems (Fiedler et al., 2018) by analyzing the impact of passenger-vehicle assignment optimality on system performance. To do this, we use a variant of the vehicle-group assignment (VGA) method used by Alonso-Mora et al. (2017) and Čáp and Alonso-Mora (2018). We chose this method because it was previously demonstrated to be able to efficiently solve large-scale DARP instances

with tight pick-up and drop-off time windows that are characteristic of large-scale MoD systems. Moreover, it is less complex than the classical exact methods for DARP, and it can be easily modified to a resource-constrained version, which we also evaluate in this work.

The contribution of this paper is 3-fold:

1. *Optimality*: We took special care to ensure that all ridesharing assignments and routes are computed optimally. This is in contrast to Alonso-Mora et al. (2017), who used a similar solution algorithm to evaluate shareability within the NYC taxi dataset, but to maintain computational tractability, the actual implementation used in the experiment resorted to heuristics and time-outs, leading to suboptimal performance of the system. Moreover, it remained unclear how far are the reported performance metrics from optimum. In this work, we identified and solved several algorithmic bottlenecks, and consequently, we were able to obtain optimal ridesharing assignments for the majority of the evaluated scenarios.
2. *Scale*: We implemented performance optimizations that enable us to significantly scale the algorithm and compute optimal ridesharing assignments for instances of unprecedented size peaking at more than 21,000 active travel requests and 11,000 vehicles. This is in contrast to Čáp and Alonso-Mora (2018) who proposed the optimal version of the VGA method but were only able to solve problem instances with a bit less than 500 requests.
3. *Impact of Assignment Optimality*: With an (1) *optimal* and (2) *scalable* implementation of a ridesharing algorithm, we are able to achieve the main objective of this work: to quantify the impact of using an optimal ridesharing method on system performance in comparison to the performance achieved by sub-optimal ridesharing methods. We quantify the reduction in vehicle distance traveled, travel delay, used vehicles, and traffic density for different ridesharing strategies. Specifically, we compare the above metrics in six scenarios: (a) a present-day transportation using private vehicles, (b) an MoD system without ridesharing, (c) an MoD system with ridesharing based on the IH, (d) an MoD system with optimal ridesharing computed using Vehicle Group Assignment (VGA) method, (e) an MoD system with ridesharing solved by a VGA method with limited computational resources, and (f) an MoD



system with ridesharing solved by a resource-constrained variant of VGA method as implemented in Alonso-Mora et al. (2017). This allows us to give a quantitative answer to the question of how much do we gain by actually taking the effort to compute optimal assignments?

The performance comparison of the system that uses optimal assignments against the system that uses IH is particularly interesting, as the latter approach is widely used in existing studies of large-scale MoD systems (Bischoff et al., 2017; Campbell & Savelsbergh, 2004; Fiedler et al., 2018), while the former represents the fundamental bound on system performance.

Our evaluation revealed that optimal ridesharing assignments can reduce the distance driven in the system by 57% compared to an MoD system without ridesharing, and simultaneously, we managed to maintain the passenger travel delay below 4 minutes. Furthermore, we found that the optimal ridesharing assignments are considerably more efficient than the assignments computed by IH. Specifically, in the system that uses optimal assignments, the total vehicle distance driven is reduced by 20%, and simultaneously, average passenger travel delay is reduced by 5%. Moreover, in order to provide insights into the limits of the VGA method, we performed a sensitivity analysis on our city-scale scenarios with respect to (1) the length of a ridesharing batch, (2) the vehicle capacity, and (3) and the maximum allowed passenger delay. Our results show that the VGA method is capable of finding optimal assignments given that vehicle capacity is no more than 5–10 passengers and the maximum allowed delay is no more than 4–7 minutes, depending on the demand structure and intensity. For scenarios using higher-capacity vehicles or with more permissive delay constraints, the VGA algorithm can no longer certify optimality of the computed ridesharing assignments.

## 2. Methodology

We use a travel demand model to generate a dataset of all private car trips in Prague. Then, we design an MoD system that can serve these existing trips with the required service quality (measured by maximum travel delay). After that, we implement the considered solution methods for passenger-vehicle matching. Finally, we simulate various scenarios in multi-agent simulation and analyze the results.

### 2.1. Input data

The set of trips that represent the transportation demand is generated by the multi-agent activity-based model described in Drchal et al. (2019). We chose the city of Prague, the Czech Republic for a case study because (a) we have access to the travel demand model for the area and (b) because its demand density, demand structure, and road topology are representative for a large European city. This is in contrast to previously considered urban areas, such as Manhattan or Singapore, which due to an extremely high density of travel demand, lead to overly-optimistic estimates of the benefits of ridesharing. Nevertheless, for interested readers, we also performed a version of our experiment using the Manhattan taxi demand dataset (the dataset previously used by Alonso-Mora et al. (2017)), and we present the results in Appendix A.

In contrast to traditional four-step demand models (Hensher & Button, 2007), which use trips as the fundamental modeling unit, activity-based models employ so-called activities (e.g., work, shop, sleep) and their sequences to represent the transport-related behavior of the population. Travel demand then occurs due to the agents' necessity to satisfy their needs through activities performed at different places at different times. These activities are arranged in time and space into sequential daily schedules. Trip origins, destinations, and times are endogenous outcomes of activity scheduling. The activity-based approach considers individual trips in context and therefore allows representing realistic trip chains.

The model used in this work covers a typical work-day in the metropolitan area of Prague. The population of over 1.3 million is modeled by the same number of autonomous, self-interested agents, whose behavior is influenced by their sociodemographic attributes, current needs, and situational context. Individual decisions of the agents are implemented using four modules responsible for choosing the activity type, duration, location, and mode. Each module uses a dedicated machine learning model (such as neural network, decision trees, regression tree) trained so that its output matches various real-world data sets such as travel diaries and other transportation-related surveys, demographic data, points of interests, and transport network structure. Planned activity schedules are simulated and tuned, and finally, their temporal, spatial, and structural properties are validated against additional historical real-world data (origin-destination matrices and surveys) using the six-step validation framework VALFRAM (Drchal et al., 2015, 2016).

The model consists of over three million trips by all modes of transport in one 24-hour scenario, out of

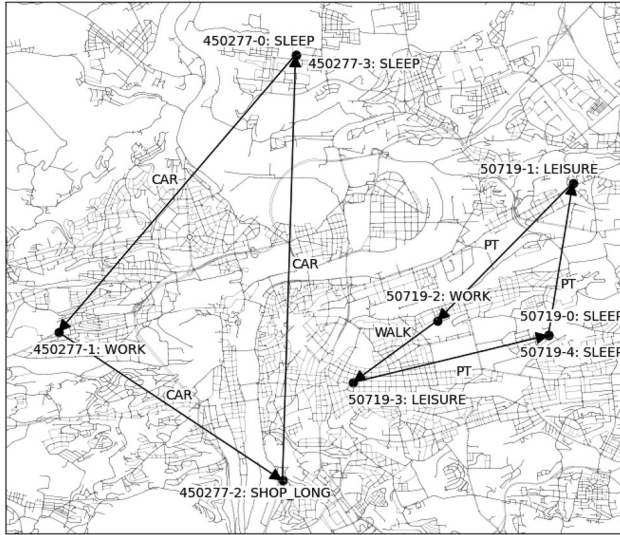
**Table B1.** Example trips.

Trip	Person	From	To	Mode
4,500,942	50,719	0	1	PT
4,500,943	50,719	1	2	PT
4,500,944	50,719	2	3	WALK
4,500,945	50,719	3	4	PT
4,789,903	450,277	0	1	CAR
4,789,904	450,277	1	2	CAR
4,789,905	450,277	2	3	CAR

Each trip connects two activities, shown in Table B2. We can identify each activity by Person column and From/To columns that correspond to the Activity column in Table B2.

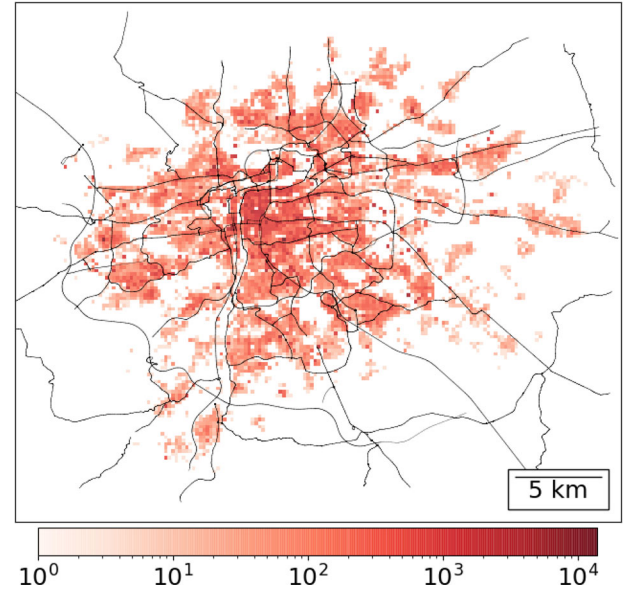
**Table B2.** Example activities.

Person	Activity	Start	End	Type	Lat	Lon
50,719	0	00:00	04:06	SLEEP	50.084294	14.490635
50,719	1	06:56	07:42	LEISURE	50.110286	14.496852
50,719	2	10:31	13:49	WORK	50.086623	14.461201
50,719	3	15:01	16:04	LEISURE	50.076027	14.439032
50,719	4	17:17	00:00	SLEEP	50.084294	14.490635
450,277	0	00:00	07:22	SLEEP	50.131751	14.423139
450,277	1	07:54	15:43	WORK	50.084170	14.360924
450,277	2	16:00	16:35	SHOP_LONG	50.059205	14.420547
450,277	3	17:26	00:00	SLEEP	50.131751	14.423139

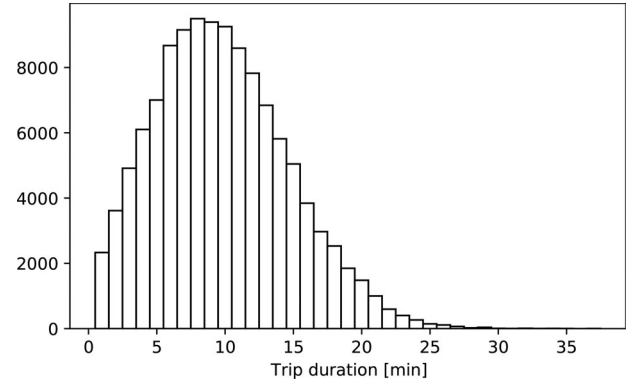


**Figure B1.** Two example trips from the generated demand. The filled circles represent activities, while the arrows represent trips between those activities. Next to each activity, we can see the person and activity IDs in the format person\_id-activity\_id. The activities corresponding to these IDs can be found in Table B2.

which there are roughly one million trips realized by private vehicles (Figure B2). Tables B1 and B2 and Figure B1 show example activity schedules for two agents. In this work, we select only the trips realized by private vehicles in two representative time intervals: the *peak* dataset includes trips that start 06:30 and 08:00, and the *off-peak* dataset includes trips that start between 10:30 and 12:00. The two datasets contain about 130,000 and 45,000 trips, respectively. The duration of trips ranges from 1 to 37 minutes; the histogram is in Figure B3.



**Figure B2.** Demand for personal vehicle traffic in Prague. The start positions of all vehicle trips are discretized to squares of 200 square meters. Darker color translates to higher demand, and the color bar has a logarithmic scale.

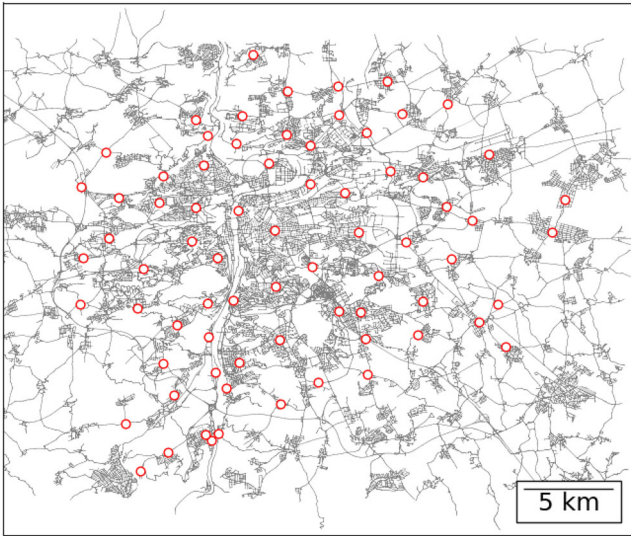


**Figure B3.** Histogram of fastest path travel times for each trip.

## 2.2. System model

For MoD systems design, we adopt a station-based methodology described by Pavone et al. (2012) or by Wallar et al. (2019), which means that idle MoD vehicles are parked in dedicated parking facilities instead of parking on-street or cruising. This setup is typical in carsharing or bike-sharing systems because curb parking would take valuable urban space, and cruising for parking would increase fuel consumption and congestion. Further, in case of electric vehicles, stations will provide charging infrastructure. Vehicles are initialized in stations and leave a station only to serve travel requests. Whenever a vehicle becomes idle, it starts driving to the nearest station to park there.





**Figure B4.** MoD system stations in the city of Prague. There are 73 stations in total, shown as red circles.

### 2.2.1. Station positioning, rebalancing, and fleet-sizing

We use 73 stations shown in Figure B4 chosen such that every node on the road network (excluding roads without travel requests such as tunnels or highways) can be reached from one of the stations within 210 s, and the number of stations is minimized. We compute the station positions using an integer program with binary variables  $s_n$  for each node  $n$  in the set of serviced nodes  $N$ , where each variable  $s_n$  indicates if there is a station at node  $n$  (1) or not (0). We minimize

$$\sum_{n \in N} s_n, \quad (1)$$

subject to

$$\sum_{n' \in P_n} s_{n'} \geq 1 \quad \forall n \in N, \quad (2)$$

where  $P_n$  is a set of nodes from which  $n$  is reachable within 210 s.

The stock of vehicles at each station is stabilized by a vehicle rebalancing process that continuously sends empty vehicles from stations with a surplus of vehicles to stations that have a shortage of vehicles. We use the rebalancing policy introduced by Pavone et al. (2012) and later evaluated by Spieser et al. (2014) in the Singapore MoD case study. In one-minute intervals, we generate an integer program for transferring vehicles from stations with more vehicles compared to the initial state to stations with fewer vehicles compared to the initial state such that the number of vehicles in each station  $s$  is kept above a corresponding threshold  $\tau_s$ , and the total length of all rebalancing trips is minimized. We experimentally determined that in order to compensate for driving vehicles, the  $\tau_s$  should be no

more than 85% of the initial number of vehicles parked in  $s$ . Also, we use only stations with at least 5% more vehicles over the initial state as source stations in order to prevent rebalancing instabilities, i.e., rebalancing flows in the opposite directions.

Our objective is to achieve full service availability during the entire experiment, i.e., every request should be served. We experimentally determined that in order to be able to serve every request during the morning peak (see Sec. 2.1) without ridesharing, the MoD system requires a total of 68,201 vehicles.<sup>1</sup> We used the same fleet size for other scenarios (off-peak, ridesharing)<sup>2</sup> as these experiments are guaranteed to require fewer or equal vehicles than the scenario without ridesharing. Specifically, to determine the number of vehicles in each station needed to ensure full service availability, we first created a dedicated vehicle for each request in the station closest to the requested pickup location. Then, we started iteratively reducing the number of vehicles by the same factor in each station until the first vehicle shortage event occurred in any station. Then, we used the vehicle counts from the last iteration without any shortage. This procedure guarantees that there is a sufficient number of vehicles to serve all requests from the nearest station.

### 2.2.2. Problem formulation

The set of all vehicles will be denoted as  $V = 1, \dots, m$ , all vehicles have the same capacity  $K$ . Travel requests are modeled as a sequence  $(t_i, o_i, d_i), (t_2, o_2, d_2), \dots$ , where  $t_i$ ,  $o_i$ , and  $d_i$  are the announcement time, origin point, and destination point of request  $i$ , respectively. The  $i$ th request is revealed only at time  $t_i$ . We obtain requests from the demand model trips simply by setting  $t$  equal to the trip start time,  $o$  equal to the trip start location, and  $d$  equal to the trip end location.

The state of a vehicle  $v$  at a particular time point encodes its current position, the set of passengers currently on-board of the vehicle, and its current plan. The plan of a vehicle is represented as a sequence of locations  $p = l_1, l_2, \dots$ , where each location  $l_i$  is either an origin location  $o_i$ , or a destination location  $d_i$  of request  $i$  that is scheduled to be serviced by the plan. A vehicle plan is *valid* only if the plan contains origin location and later destination location for each onboard passenger.

The operational cost of vehicle  $v$  when following plan  $p$  is denoted  $c(p, v)$ . For simplicity, we define  $c(p, v)$  to be equal to the distance driven by the vehicle when it follows plan  $p$ . Each plan  $p$  requires a vehicle of capacity  $\kappa(p)$ , where  $\kappa(p) \leq |p|$ . The travel delay of request  $r$  when it is served by vehicle  $v$  following plan  $p$  is computed as:

$$q_r(p, v) := (t_r^{\text{dropoff}} - t_r) - \delta_r^{\text{baseline}}. \quad (3)$$

Here,  $t_r^{\text{dropoff}}$  is the time when the request is dropped off under plan  $p$  and  $\delta_r^{\text{baseline}}$  is the duration along direct route from the request's origin to its destination. Note that the passenger's waiting time is included in the delay, and therefore, the maximum delay also limits the maximum waiting time.

Our goal is to minimize the total operational cost of the system, such that the delay of every passenger is bounded by a constant  $q_{\max}$ , and the maximum capacity  $K$  is respected for all vehicles. That is, we desire to minimize

$$\sum_{v \in V} c(p, v) \quad (4)$$

subject to

$$q_r(p, v) \leq q_{\max} \quad \forall r \in R \quad (5)$$

$$\kappa(p_v) \leq K \quad \forall v \in V. \quad (6)$$

### 2.3. Request-vehicle matching

In an MoD system, new requests dynamically arrive and need to be served. A ridesharing algorithm tries to find the *optimal system plan* (i.e., a collection of vehicle plans), such that (1) every request is served, (2) maximum discomfort constraint  $q_{\max}$  is respected, and (3) the total operation cost is minimized. This planning procedure is repeated periodically, and each such planning period is referred to as a *batch*. During one batch, we collect all newly announced requests and execute a planning procedure that computes request-vehicle matching and corresponding vehicle plans. We make the following assumptions: (a) travel time on each road segment is constant over time and does not depend on the number of vehicles on the segment, (b) the execution of the vehicle schedule is perfect (there are no random delays), and (c) the mode choice is fixed in the demand model and customers accepts any plan that satisfies the max delay constraint (which is guaranteed in our setup, as explained in Sec. 2.2.1).

The request-vehicle matching can be modeled as a Dial-a-Ride (DARP) problem, which is known to be NP-hard (Toth & Vigo, 2014). In this work, we implement and compare two methods for computing such request-vehicle matching. First, we implement Insertion Heuristic (IH) (Campbell & Savelsbergh, 2004), a popular heuristic algorithm for DARP and other vehicle routing problems. Second, we implement Vehicle-Group Assignment (VGA) method (Čáp & Alonso-Mora, 2018), which is a recently proposed exact solution method for DARP exhibiting good scalability properties.

#### 2.3.1. Insertion heuristic

The pseudocode of the IH is presented in Algorithm 1. The algorithm is implemented as follows: For each new request, the IH algorithm attempts to insert the request into the plan of each vehicle. The current plan of a vehicle  $v$ , denoted as  $p_v$ , is the plan computed in one of the previous iterations of the algorithm. For a particular vehicle  $v$ , we try all possible indexes  $i$  in plan  $p_v$  to insert pickup of the new request before  $i$  and all possible indexes  $j, j > i$  to insert drop off of the new request before  $j$ . We denote such plan as  $p_v^{\text{new}}$ . Note that the relative ordering of all locations from  $p_v$  remains unchanged in the new plan, and therefore, optimality is not guaranteed. Finally, among all plans generated this way, we select the plan (and the corresponding vehicle) that minimizes the increase in operating cost and at the same time satisfies the service discomfort constraints.

---

#### Algorithm 1: Insertion Heuristic

---

**input:** Current plan  $p_v$  of each vehicle  $v$  that was computed in one of the previous iterations of the algorithm and the set of new requests  $D_n$ ,  
i.e., requests announced in the  $n$ th batch.

```

1  for  $r \in D_n$  do
2     $\delta_c^{\min} \leftarrow \infty$ ; /* min. cost increment */
3     $v^* \leftarrow \text{null}$ ;
4    for  $v \in V$  do
5      for  $i \in 1, \dots, |p_v|$  do
6        for  $j \in i + 1, \dots, |p_v| + 1$  do
7           $p_v^{\text{new}} \leftarrow p_v$ ;
8          insert  $o_r$  to  $p_v^{\text{new}}$  before index  $i$ ;
9          insert  $d_r$  to  $p_v^{\text{new}}$  before index  $j$ ;
10          $\delta_c \leftarrow c(p_v^{\text{new}}) - c(p_v)$ ;
11         if  $p_v^{\text{new}}$  is feasible and  $\delta_c < \delta_c^{\min}$  then
12            $\delta_c^{\min} \leftarrow \delta_c$ ;
13            $p^* \leftarrow p_v^{\text{new}}$ ;
14            $v^* \leftarrow v$ ;
15         if  $v^*$  not null then
16           vehicle  $v^*$  follows plan  $p^*$ 

```

---

#### 2.3.2. Vehicle group assignment method

The VGA method relies on the performance improvement coming from conversion of a DARP problem to a variant of assignment problem. In this work, we generalize the formulation by Čáp and Alonso-Mora (2018) to be applicable in an online optimization setting. That is, we reformulated the algorithm to support optimization with requests already onboard some vehicles because the methodology by Čáp and Alonso-

Mora (2018) assumes all vehicles to be empty before the request-vehicle matching.

The VGA method can be divided into two phases: group generation (Algorithm 3) and vehicle-group assignment (Problem 1). We can see the overall pseudo-code in Algorithm 2. Let  $D_v$  be a set of waiting requests, i.e., the set of requests that have not been picked up yet. Further, let *group* be a set of requests such that for each group  $R$ ,  $R \subseteq D_w$ . In the first phase, for each vehicle, we compute all groups that can be serviced by the vehicle without violating the maximum delay  $q_{\max}$  using the *group generation* algorithm (Algorithm 3). The second phase uses ILP (Problem 1) to map exactly one group to each vehicle so that every request is serviced and the system plan is optimal. The whole procedure is demonstrated by an example in Figure B5.

---

**Algorithm 2:** VGA method

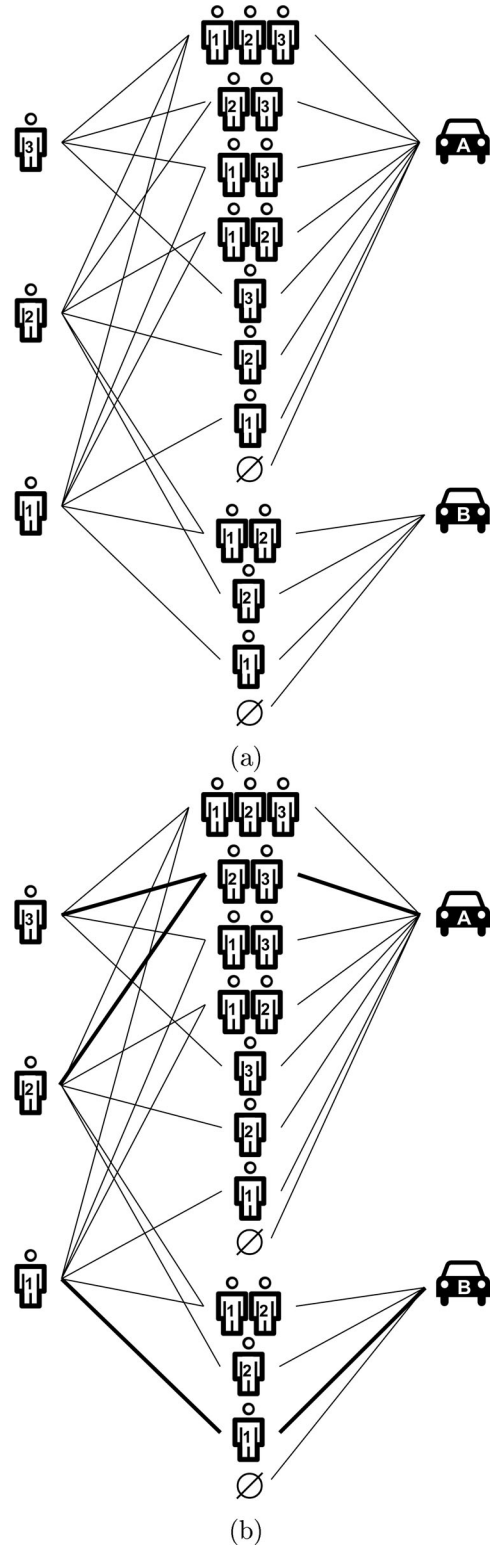
---

**input:** The current position and on-board passengers for each vehicle in  $V$  and the set of waiting requests  $D_w$ .

- 
- 1 **for**  $v \in V$  **do**
  - 2      $\Gamma_v \leftarrow \text{generate\_groups}(v, D_w)$ ;
  - 3      $\pi^* \leftarrow \text{Solve Problem 1 using } \Gamma_1 \dots \Gamma_m$ ;
  - 4     All vehicles follow the optimal system plan  $\pi^*$ ;
- 

We say that a group  $R$  is *feasible* for vehicle  $v$  if a plan exists for the vehicle that serves all requests from  $R$  without violating maximum delay and capacity constraints and if all requests onboard vehicle  $v$  are members of the group. We denote a set of all groups feasible for vehicle  $v$  by  $\Gamma_v$ . The key property of feasible groups, observed by Alonso-Mora et al. (2017), is that if a group  $R_1$  is feasible, all subsets of the group  $R_2 \subset R_1$  are also feasible. This structural property is used to limit the number of groups we need to test for feasibility in the first part of the VGA method, the group generation algorithm. To determine if a group is feasible, we define function  $f(R, v)$  that indicates whether the group  $R$  is feasible for vehicle  $v$ .

The group generation algorithm (Algorithm 3) computes feasible groups for each vehicle independently. First, the group generation algorithm computes all feasible requests for each vehicle  $v$ , i.e., the feasible groups of size 1 marked as  $\Gamma_v^1$  (lines 4–6). Then, we find larger groups iteratively by combining the feasible groups from the previous iteration with all feasible requests (lines 7–14). From the performance perspective, it is important to try each group only once (the purpose of *checked*) and also to check



**Figure B5.** Example of the VGA method assigning three passengers to two vehicles. (a) All possible request groups for each vehicle. The lines between the request (left) and the group (middle) denote the membership in the group. The lines between the groups and vehicles denote feasible group assignments. (b) Final assignment between vehicles and group is shown (bold lines).

that all possible subsets of  $R$  of size  $|R| - 1$  are present in  $\Gamma_v^{|R|-1}$  before checking group  $R$  for feasibility. At the end of this step, we have a set of feasible groups for each vehicle, as it is illustrated in Figure B5(a).

**Algorithm 3:** Function `generate_groups` that generates groups for vehicle  $v$ . The Boolean-valued function  $f(R, v)$  evaluates to true if vehicle  $v$  can serve all requests from group  $R$  without violating maximum delay and capacity constraints.

---

**input:** A vehicle  $v$  and the set of waiting requests  $D_w$ .  
**output:** Set of all feasible groups for the vehicle ( $\Gamma_v$ ).

---

```

1  Let  $R^{\text{init}}$  be the set of requests onboard vehicle  $v$ ;
2   $k \leftarrow \max(|R^{\text{init}}|, 1)$ ;
3   $\Gamma_v^k \leftarrow \{R^{\text{init}}\}$ ;
4  for  $r \in D_w$  do
5      if  $f(\{r\}, v)$  then
6           $\Gamma_v^1 \leftarrow \{\{r\}\} \cup \Gamma_v^1$ ;
7  while  $\Gamma_v^k \neq \emptyset$  do
8       $\Gamma_v^{k+1} \leftarrow \emptyset$ ;
9      /* not check groups repeatedly */
10     checked  $\leftarrow \emptyset$ ;
11     forall  $R \in \Gamma_v^k, \{r\} \in \Gamma_v^1$  do
12         if  $(R \cup \{r\}) \notin \text{checked}$  and  $\forall R' \subset (R \cup \{r\}), |R'| = k : R' \in \Gamma_v^k$  and  $f(R \cup \{r\}, v)$  then
13              $\Gamma_v^{k+1} \leftarrow (R \cup \{r\}) \cup \Gamma_v^{k+1}$ ;
14             checked  $\leftarrow \text{checked} \cup (R \cup \{r\})$ ;
15      $k \leftarrow k + 1$ ;
16 if  $|R^{\text{init}}| > 0$  then
17      $\Gamma_v^1 \leftarrow \emptyset$ ;
18  $\Gamma_v \leftarrow \{\emptyset\} \cup \Gamma_v^1 \cup \Gamma_v^2 \cup \dots \cup \Gamma_v^k$ ;

```

---

The second part of the method finds the assignment of groups to vehicles that minimizes the total traveled distance resulting from vehicle plans such that for each vehicle, exactly one of the groups feasible for the vehicle is assigned, and all requests are served. The assignment of groups to vehicles is formulated as an ILP. There is a binary variable  $\xi_v^g$  for each possible vehicle-group assignment where  $\xi_v^g = 1$  if a group  $g \in \{1, \dots, |\Gamma_v|\}$  is assigned to vehicle  $v$  and  $\xi_v^g = 0$  otherwise. Using these variables, the problem is defined as:

**Problem 1** (Vehicle-group Assignment)

$$\min \sum_{v=1}^m \sum_{g=1}^{|\Gamma_v|} \xi_v^g c(p_v^{g*}),$$

subject to

$$\sum_{g=1}^{|\Gamma_v|} \xi_v^g = 1 \quad \forall v \in V \quad (7)$$

$$\sum_{v=1}^m \sum_{R=1}^{|\Gamma_v|} \mathbf{1}_R(r) \xi_v^R = 1 \quad \forall r \in D_w \quad (8)$$

In the objective function,  $p_v^{g*}$  denotes the optimal plan for vehicle  $v$  to serve group  $R_g$ . Constraint (7) states that only one group can be assigned to each vehicle. Constraint (8) ensures that each request is served by exactly one vehicle plan. The indicator function  $\mathbf{1}_{R_g}(r)$  is equal to 1 if the request  $r$  is a member of the group  $R_g$  and 0 otherwise.

By solving the above described ILP, we obtain an optimal assignment of vehicles to feasible groups (see Figure B5b for example assignment). This assignment can be directly translated into vehicle plans that replace vehicle plans from the previous iterations.

### 2.3.3. Complexity, optimality, and implementation

The worst-case complexity of computing an assignment of a set of waiting requests  $D_w$  to a set of vehicles  $V$  using IH is  $O(|D_w| \cdot |V| \cdot l)$  where  $l$  is the length of a plan and can be bounded as  $l \leq K + |D_w|$ . For VGA, we analyze the complexity of each phase separately. For group generation, the computational complexity is dominated by the need to verify the feasibility of a group, represented by the call of function  $f(R, v)$ . Solving this function, in fact, equals solving a single-vehicle DARP (see Section B for more detail), which is an NP-hard (Toth & Vigo, 2014) problem. The vehicle-group assignment is then obtained by solving an ILP, which is also, in general, an NP-hard problem (Schrijver, 1986). Therefore, the VGA method can, in the worst case, require computational time that is exponential in the number of requests and vehicles. However, DARP problem instances appearing in the context of large-scale MoD systems tend to have structural properties that are beneficial to the VGA algorithm. Specifically, since large-scale MoD systems are designed to provide quality of service comparable to using a private vehicle, the maximum waiting at pick-up is usually constrained to be less than a few minutes, and similarly limited is the maximum delay at destination. Such tight pick-up and drop-off time window constraints are used by the VGA algorithm to prune the feasible solution space. In practice, the maximum group size that requires a feasibility check tends to be relatively small, and also the total number of feasible groups tends to be within the grasp of existing ILP solvers. Under such conditions, the



VGA algorithm is able to generate optimal results in practical computation time.

As for the optimality, IH is a heuristic approach, and as such, it cannot guarantee that the generated solution is optimal. VGA method will generate an optimal solution if all feasible groups are generated exhaustively, and the ILP program is solved to optimality. For proof, see Čáp and Alonso-Mora (2018).

For our case study, we implemented the IH and the VGA method in Java. The ILP appearing in the VGA method is solved using Gurobi.<sup>3</sup> The request-vehicle matching procedure is run every 30 s of the simulation for both IH and VGA. For both methods, the maximum delay constraint is set to 4 min, and the vehicle capacity is set to five passengers. The ILP solver in the VGA method computes the optimal solution with the maximum optimality gap of 0.02%.

While the ability to compute optimal ridesharing assignments is essential to understand the limit of performance gains that can be achieved by ridesharing (and the gap between the optimal performance and the performance of the heuristic solutions), a practical deployment may impose constraints on the maximum run time of a ridesharing algorithm. Therefore, we also tested a resource-constrained version of the VGA method with the ILP solver optimality gap set to 0.5% and group generation time-limited to 60 ms per vehicle. We refer to this version as *VGA limited*. Also, for comparison, we reimplemented the method proposed by Alonso-Mora et al. (2017) in their Manhattan taxi ridesharing study. As described in the supplemental material of Alonso-Mora et al. (2017), this solution method employs specific heuristics and optimization cutoffs to achieve practical run time. We refer to this version as *VGA PNAS*.

We compute passenger-vehicle assignments together with the simulation sequentially, and thus from a simulation perspective, the ridesharing computation is an instantaneous event. In case of practical deployment, one could achieve sufficiently low wall-clock running time by computing on a computational cluster with many CPU cores because the VGA algorithm is easily parallelizable.

The existing variants of the VGA method (Alonso-Mora et al., 2017; Čáp & Alonso-Mora, 2018) were not able to solve the ridesharing instances appearing in our case study to optimality within 24 hour limit on computational time or with a 60 GB memory limit. Therefore, we implemented performance optimization described in Appendix B that enabled us to find optimal solutions to our instances respecting these limits.

## 2.4. Simulation

In our experiments, we simulated the following five scenarios:

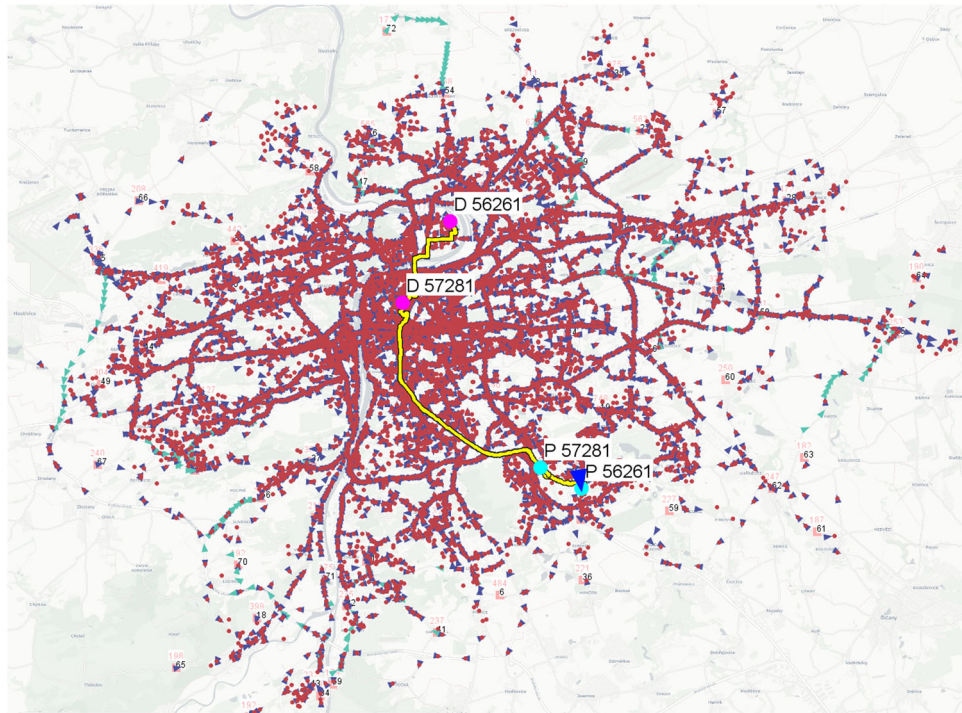
- *Present state*: All the requests are served by private vehicles. The vehicles are parked at the request's start location, i.e., there is no delay. The number of used vehicles is equal to the number of requests, and the total distance traveled is equal to the sum of the shortest paths between the origins of all requests and their destinations.
- *MoD w/o ridesharing*: MoD system without ride-sharing, the plans are computed using IH, and the vehicle capacity is set to one, i.e., the passengers are not allowed to share rides.
- *MoD w. IH Ridesharing*: MoD system with ride-sharing computed by the IH.
- *MoD w. VGA Ridesharing (optimal)*: MoD system with ridesharing computed by the VGA method to optimality.
- *MoD w. VGA Ridesharing (runtime limited)*: MoD system with ridesharing computed by the VGA method, with the group generation time-limited to 60 ms per vehicle and the ILP solver maximum optimality gap of 0.5%.
- *MoD w. VGA Ridesharing (PNAS)*: MoD system with ridesharing computed by the VGA method, with a set of timeouts/heuristics as described in Alonso-Mora et al. (2017), which we have reimplemented for this article.

We simulate a morning peak time interval 7:00–8:00 and an off-peak time interval 11:00–12:00. To avoid the “cold start” artifacts, the simulation begins 30 minutes before the analyzed time interval, at 6:30 and 10:30, respectively, but for subsequent analysis, we only use the data captured after the thirty-minute start period. Including the 30 minute warm-up time, there are 122,473 requests in the morning peak, and 42,633 requests in the off-peak experiment.

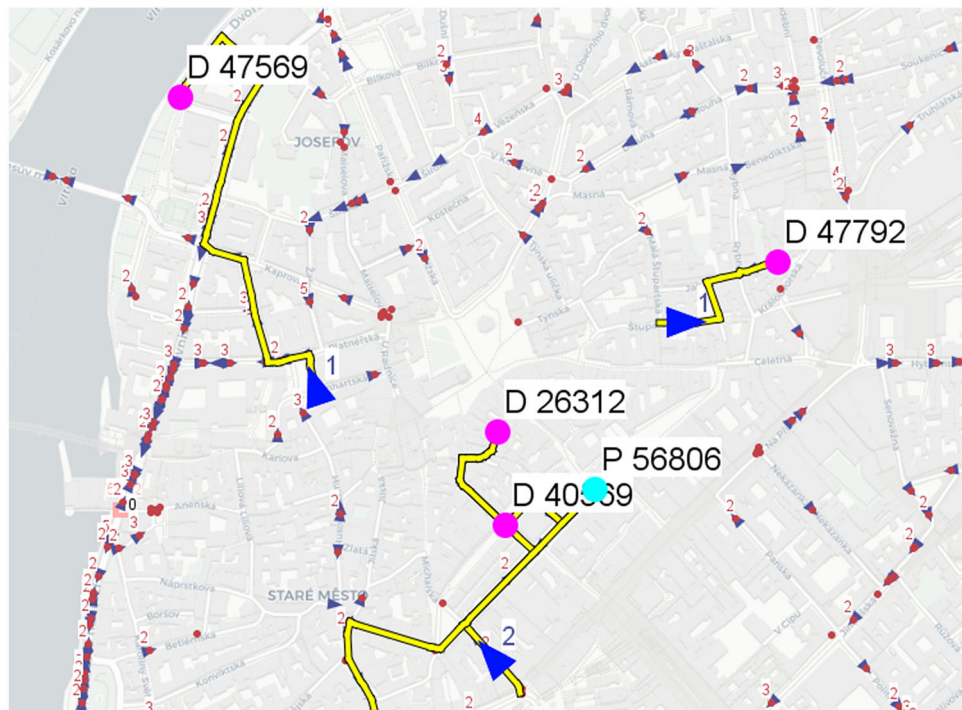
The scenarios were simulated in the multi-agent transportation simulation framework AgentPolis.<sup>4</sup> The simulation environment consists of a) road network composed of *nodes* (crossroads) and *edges* (road segments), b) on-demand vehicle stations, c) on-demand vehicle agents, and d) passenger agents. In Figure B6, we show a screenshot of the AgentPolis visualization captured during one of the simulation experiments.

We use an OpenStreetMap<sup>5</sup> road network consisting of 158,674 edges and 63,995 nodes. The speed limit for each road segment was also taken from OpenStreetMap data, and missing entries were generated according to





(a)



(b)

**Figure B6.** AgentPolis visualization of the simulated traffic in Prague during the traffic peak. (a) (Left): The entire city of Prague in the simulation. A more detailed (zoomed in) view can be seen in (b) (right). Vehicles are represented as blue triangles, with a number indicating the onboard passenger count. Red circles represent passengers. Some vehicles are highlighted, and their current plan is drawn with a yellow line. The pick-up and drop off locations of the remaining actions are marked with cyan and pink circles, respectively, with a number indicating passenger ID. Note that in (b), there are some passengers already driving in two of the vehicles, so the number of drop-off locations is greater than the number of pick-up locations. The green triangles are vehicles that travel empty between stations (rebalancing). Video: <https://sum.fel.cvut.cz/agentpolis/>

following rules based on the local legislation: highway: 130 km/h, living street: 20 km/h, otherwise: 50 km/h.

During initialization, we create vehicle stations, each filled with the pre-determined number of vehicles. During the simulation, we are creating passenger agents for each request at its announcement time and origin point. Each passenger is then picked up by the assigned on-demand vehicle, driven to the desired location, dropped off, and finally released from the simulation. The vehicle to serve the passenger is selected using the passenger-vehicle matching procedure (see Sec. 2.3), either IH or VGA. Note that each passenger can be either matched to one of the empty vehicles parked in a station or to a vehicle already serving some previously assigned requests. Each vehicle executes its plan until it becomes empty (i.e., all assigned passengers have been dropped off), then it drives to park itself in the nearest station. As explained in our assumptions, the passengers always select the MoD system as the mode for their trip. In the simulation, any request that would wait for longer than 4 minutes or would be delivered to its destination with more than 4 minutes of delay is considered a rejected request. However, as mentioned before, we configured the system so that these quality of service bounds are always satisfied, and consequently, there are no rejections during the simulation experiment.

### 3. Results

In this section, we present the simulation results. To run the experiments, we used a desktop system with Intel Core i7-8700K CPU (3.7 GHz, 6/12 physical/virtual cores) and 64 GB RAM.

#### 3.1. Operating cost and computational time

Tables B3 and B4 summarize the main results of the experiments. As explained in Sec. 2.2, we computed the size of the fleet to always guarantee full service availability. Since the service level is always 100%, we do not show this metric in result tables and plots. The first row shows the value of our optimization criterion, i.e., the system operation cost measured in terms of total distance driven by the fleet vehicles. We can see that when using the VGA method instead of IH during the morning peak, we can save more than 110,000 km of vehicle distance driven, which represents more than 20% reduction. Compared to the “no ridesharing” scenario and to the present state, the VGA method saves over 573,000 km (57%) and 328,000 km (43%), respectively. Even in off-peak time, the VGA method can save about 17% of the total distance driven compared to the IH, and about 48% compared to the “no ridesharing” scenario.

The VGA method is considerably slower than IH. The average computational time per one optimization batch in the peak scenario was about 193 s, compared

**Table B3.** Main results from the considered scenarios during the morning peak (7:00–8:00).

Optimal	Present	Mobility-on-demand				
		No Ridesh.	IH	VGA	VGA lim	VGA PNAS
	–	–	no	yes	no	no
<b>Total veh. dist. (km)</b>	<b>758,001</b>	<b>1,002,766</b>	<b>539,793</b>	<b>429,172</b>	<b>451,978</b>	<b>475,378</b>
Avg. delay (s)	–	132	190	180	178	161
Avg. density (veh./km)	0.0077	0.0085	0.0053	0.0046	0.0048	0.0049
Congested seg.	8	25	1	1	1	0
Heavily loaded seg.	163	291	31	10	17	20
Used vehicles	122,473	33,066	15,685	13,787	14,449	14,607
Avg. comp. time (ms)	–	181	18	192,903	27,714	15,598

Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with density above 50% of the critical density.

**Table B4.** Main results from the considered scenarios, off-peak (11:00–12:00).

Optimal	Present	Mobility-on-demand				
		No Ridesh.	IH	VGA	VGA lim	VGA PNAS
	–	–	no	yes	no	no
<b>Total veh. dist. (km)</b>	<b>283,483</b>	<b>344,613</b>	<b>211,285</b>	<b>175,865</b>	<b>176,957</b>	<b>186,520</b>
Avg. delay (s)	–	131	191	179	179	165
Avg. density (veh./km)	0.0045	0.0047	0.0034	0.0032	0.0032	0.0032
Congested seg.	0	1	1	0	0	0
Heavily loaded seg.	5	10	3	1	2	2
Used vehicles	42,633	7727	4646	4746	4802	5180
Avg. comp. time (ms)	–	4	1	5438	4408	4113

Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with density above 50% of the critical density.

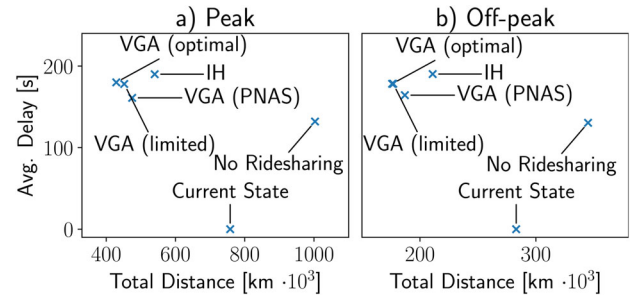
to 18 ms for the IH. Such a difference in the computational time may look extreme, but we have to consider the scale of the scenarios that were solved to optimality using the VGA method. The largest assignment problems (batches) contained more than 3000 waiting requests, 21,000 active requests (including passengers already driving to their destination), and 11,000 vehicles.

The runtime-limited experiment shows that we can speed up the VGA method significantly by merely limiting the computational time for the group generation and the solver. In the VGA limited experiment, we reduce the computation time more than six-fold over the unconstrained version of the VGA method while still reducing the total traveled distance by more than 16% over the IH. The VGA PNAS experiment reduces the computational time by another 42% at the cost of being closer to IH in the traveled distance (12% improvement). In the off-peak scenario, the VGA limited performs almost the same as the unconstrained version because the time limits are rarely reached. Note, however, that there is more than a 5% increase of traveled distance in VGA PNAS, despite similar computational times suggesting that this method is not suitable for scenarios where sufficient computational resources are available.

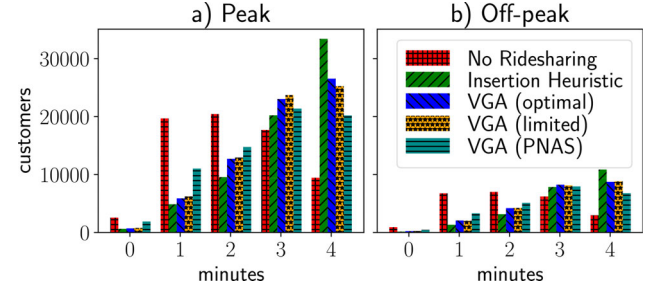
### 3.2. Tradeoff between operating cost and passenger discomfort

Another metric that we tracked is the service quality, represented by the passenger delay relative to transportation by the private vehicle. From Tables B3 and B4, we can see that the optimal VGA method saves about 5% time over the IH in both peak and off-peak experiments. The tradeoff between the operating cost (distance traveled) and the service quality (average delay) is depicted in Figure B7.

A more detailed overview of the passenger delays with a delay histogram for the four MoD scenarios in both time windows is in Figure B8. It is clear that for both peak and off-peak time, the VGA method reduces the passenger delay resulting from ridesharing compared to the IH. Nevertheless, even in the case of the VGA method, there is a noticeably greater delay compared to the no ridesharing scenario, where the delay can occur only before the passenger is picked up or over the present state, where there is no delay because a car is assumed to be available at the origin of each passenger trip.



**Figure B7.** The tradeoff between total distance traveled by all vehicles and the average delay of one passenger trip.



**Figure B8.** Histograms of delays. The present state scenario is omitted as the delay is always zero.

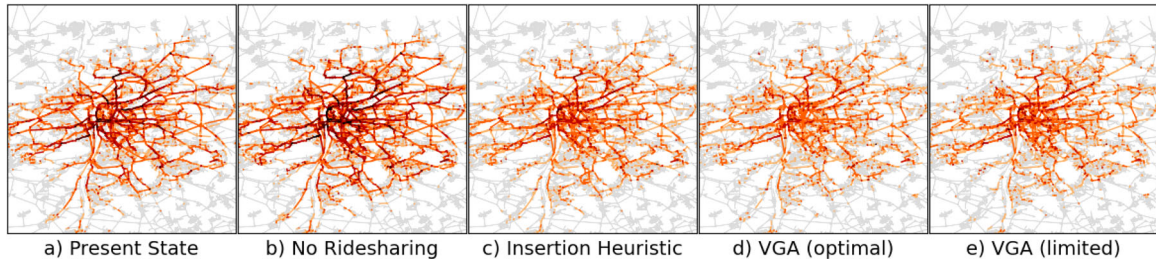
### 3.3. Impact of MoD on congestion

In addition to the operational cost, we measured the impact of the MoD system on congestion. We consider road segments with traffic density above the critical density of  $0.08 \text{ vehicle m}^{-1}$  (Tadaki et al., 2015) as *congested*. Segments with density above  $0.04 \text{ vehicle m}^{-1}$  are considered as *heavily loaded*. As you can see in Table B3, in the morning peak, using the optimal VGA method reduces the average traffic density by 13% over the ridesharing that uses IH, and by 46% and 40% over the MoD without ridesharing and the current state, respectively. We can see the same trend when we look at the number of congested and heavily loaded segments. In the off-peak experiment, the situation is similar, but the absolute numbers indeed show that there is no congestion in any of the scenarios. Finally, Figures B9 and B10 depict traffic densities on every road for all five scenarios.

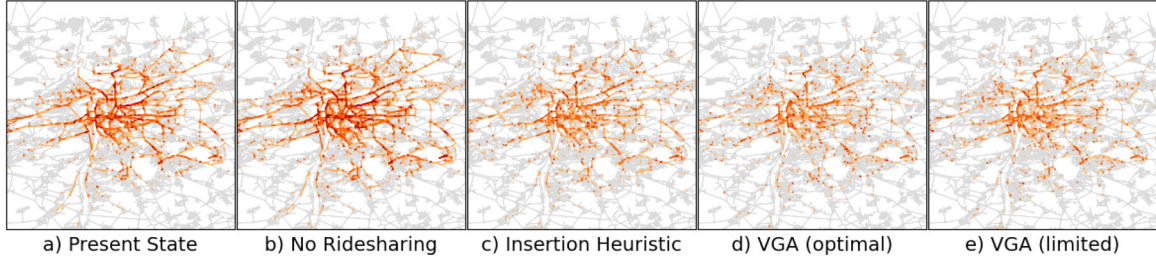
### 3.4. Fleet size and vehicle occupancy

Also, for each scenario, we recorded the number of vehicles that were used at least once during the simulation. For the present state scenario, we consider a dedicated vehicle for each request. Therefore, the number of used vehicles is equal to the number of requests. The results confirm that the VGA method indeed makes the MoD system more efficient. During peak-hour, the optimal VGA used 1898 (12%) fewer

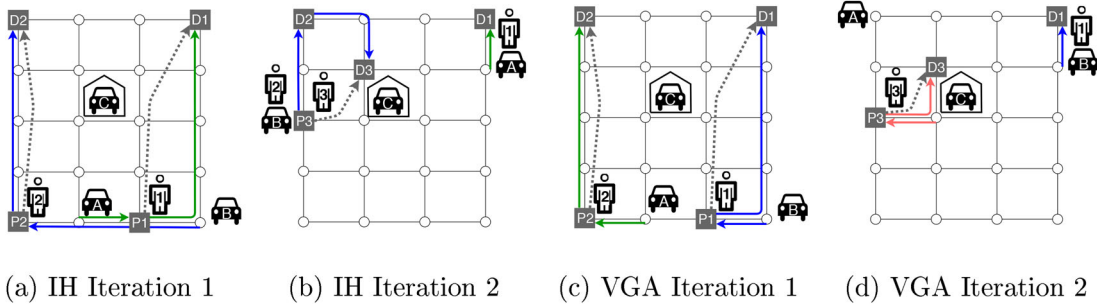




**Figure B9.** Traffic density map of the four scenarios during the morning peak. Darker colors signalize higher traffic density. Black color means that the road segment is congested. We omit the density map for the VGA PNAS experiment from this figure as it is very similar to the density map for the VGA limited experiment.



**Figure B10.** Traffic density map of the four scenarios during the off-peak time. Darker colors signalize higher traffic density. Black color means that the road segment is congested. We omit the density map for the VGA PNAS experiment from this figure as it is very similar to the density map for the VGA limited experiment. (a) IH Iteration 1, (b) IH Iteration 2, (c) VGA Iteration 1, (d) VGA Iteration 2.



**Figure B11.** Example of the capital cost paradox. (a, b) Two iterations of the IH. (a) There are three vehicles: vehicles A and B, and vehicle C that resides in the station, representing a potentially unlimited pool of vehicles. Also, there are two passengers (1 and 2), that request the travel from their current locations P1 and P2 to their destinations D1 and D2 (denoted by dashed arrows). Solid arrows denote the plans for both vehicles computed by the first iteration of the IH. (b) There is the same scenario in the next iteration. Both cars moved by five steps in the grid, and also, a new request appeared. We can see the new plans generated by the second iteration of IH too. The second set of figures (c and d) shows the exact same two iterations solved by the VGA method. Note that although we saved one segment of traveled distance (vehicles traveled 14 segments in the grid combined compared to 15 segments in case of the IH), we used one extra vehicle (vehicle C) that was not needed in the IH scenario, thus effectively increased the required fleet.

vehicles than the IH. Compared to the MoD system without ridesharing, the MoD system with optimal ridesharing used about one-third of the vehicle fleet, and compared to the present state system, the reduction is almost thirteen-fold.

In the off-peak time, however, we registered that the optimal VGA method uses about 2% more vehicles than IH. By analyzing the simulation output, we found an explanation for this perhaps surprising

result. First, counterintuitively, it is possible that a suboptimal vehicle assignment that generates plans with longer total distance can lead to fewer vehicles being used, as it is illustrated in Figure B11. Second, by analyzing the vehicle trips in both IH and VGA scenario, we found that such situations occur frequently due to unbalanced demand. In other words, the optimal method uses more vehicles not despite, but because its plans are more operating cost-efficient:

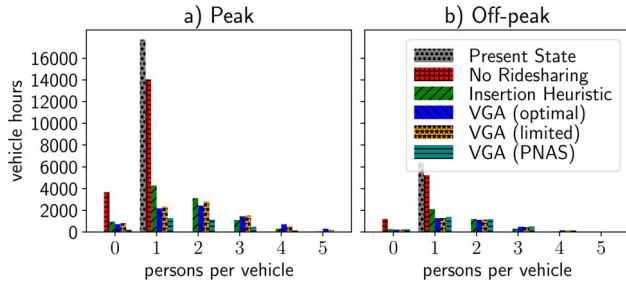


Figure B12. Occupancy histogram of all five scenarios.

the vehicles simply serve requests too quickly, which increase the chance of ending up in the areas with lower demand, where they need to wait a long time before another request appears nearby. This reminds us that to fully understand MoD systems, we need to study not only operation cost vs. service quality trade-offs, but also operation-cost vs. capital cost tradeoffs associated with different design and control strategies.

Next, we measured vehicle occupancy: Figure B12 shows the occupancy histogram for the four compared scenarios. We can see that vehicle occupancy is the highest when using the optimal method in both peak and off-peak scenarios.

### 3.5. Sensitivity analysis of the VGA method

We analyze the sensitivity of IH and the three variants of the VGA method (optimal, limited, PNAS) to variation in batch length, maximum delay, and capacity with respect to total traveled distance, computation time, and average passenger delay. Note that the time between a request announcement and the end of the batch, when the passenger-vehicle assignment is recomputed, counts toward the delay of the request. Therefore, for scenarios with longer batches, we also extended the maximum delay in order to keep the average effective maximum delay of 4 minutes. Also, note that we use the same stations and fleet for all experiments, and consequently, some requests were rejected in configurations with shorter maximum delay or longer batch length. However, the service level remains above 99% in all configurations, so the impact of rejected requests on the results is negligible.

We can see the results in Figures B13 (peak scenario) and B14 (off-peak scenario). In the peak scenario, we were able to compute the optimal solution only for a batch length of up to 30 seconds and for a maximum delay of up to 4 minutes. For larger values, the algorithm failed to terminate within 24 hours. As expected, the runtime of the optimal method grows exponentially with maximum delay. This is best seen in the case of the off-peak experiment, where the

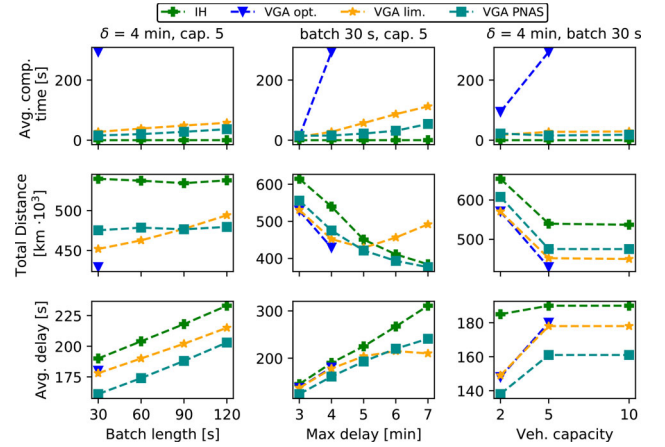


Figure B13. Sensitivity analysis: peak. Each column represents one experiment set, and inside each column, each value on the x-axis represents one experiment. Each row displays a single measured quantity. The optimal VGA method is only computed for batch length 30 s, maximum delay 3 and 4 minutes, and capacity 2 and 5 persons per vehicle. For other parameter values, the optimal VGA method did not terminate within 24 hours.

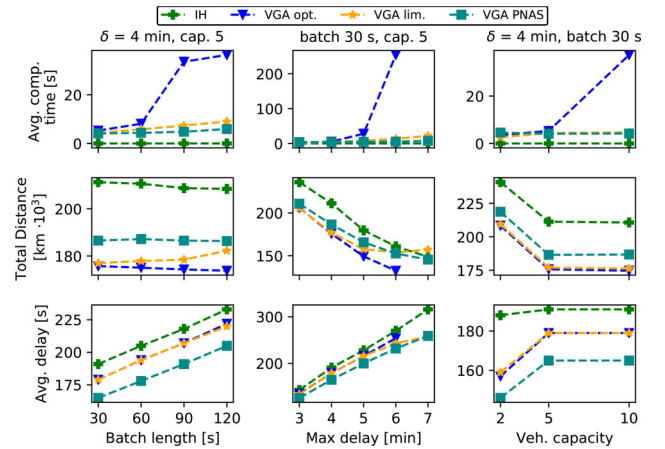


Figure B14. Sensitivity analysis: off-peak. Each column represents one experiment set, and inside each column, each value on the x-axis represents one experiment. Each row displays a single measured quantity. The optimal VGA method is only computed for the maximum delay of up to 6 minutes. For the maximum delay of 7 minutes, the optimal ridesharing assignment cannot be computed within 24 hours limit.

algorithm was able to find an optimal solution within 5 minutes of runtime on average for the 6-minute maximum delay but failed to compute optimal solutions within 24 hours runtime for the 7-minute maximum delay. Clearly, the optimal method would not scale to scenarios with larger limits on maximum delay, and one of the resource-limited variants would have to be employed.

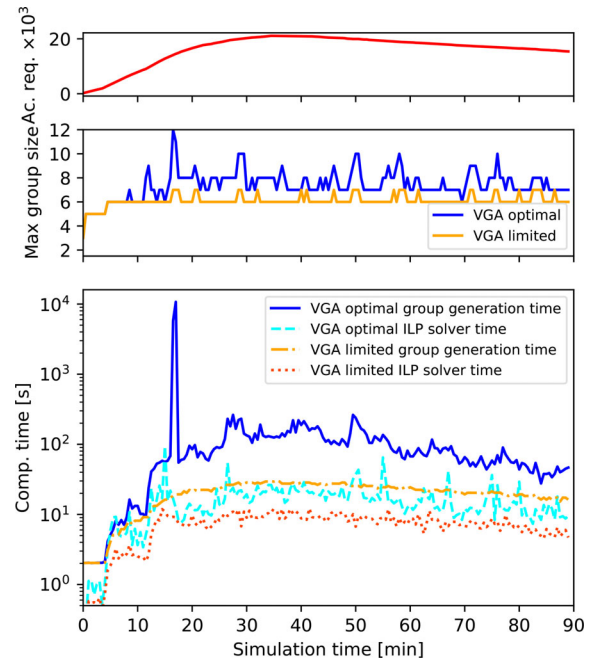
We can observe that the system's efficiency (measured in terms of total distance driven) monotonically increases with maximum allowed delay for all



considered methods with the exception of the VGA limited method, which achieves low runtimes by prematurely terminating computation in several stages of the algorithm. As we can see, the values of cutoff parameters that work well for 3–5 minute maximum delay lead to inferior performance for larger maximum delays. We can also observe that with the increasing maximum delay, the gap between the optimal method and both resource-constrained VGA methods increases. Remarkably, our experiments show that for high values of max delay, the IH achieves almost identical performance as the VGA PNAS algorithm while using only a fraction of computational resources.

The batch length negatively impacts the average travel delay experienced by passengers because they need to wait for the end of the batch for their request to be assigned to a vehicle. The motivation for using longer batch lengths is to gather more requests and to find a more efficient passenger-vehicle assignment. However, it appears that even in the off-peak experiment, these efficiency gains get only realized using the optimal solution method. For suboptimal solution methods, the efficiency gains are either negligible or straight-out negative. The higher vehicle capacity also increases the potential for ridesharing, which, in turn, could improve the operational efficiency of the system. In the peak experiment, the increased complexity prevented the computation of the optimal solution for vehicle capacity set to 10 passengers. However, our results show that the resource-limited variants were not able to improve the solution significantly. This can be caused by reaching the time limits before the algorithm can generate high-occupancy plans, or it can indicate that the capacity of 5 passengers per vehicle is sufficient. For reference, in the off-peak experiment, we were able to compute optimal solutions, and our results show that the vehicle capacity of 5 is sufficient for the off-peak demand intensity. However, it is still possible that high-capacity vehicles would be better utilized when planning for peak-intensity demand.

Concerning the suboptimal versions of the VGA method, our VGA limited method computes higher quality solutions for shorter batch lengths and shorter maximum travel delays, while the PNAS version of the VGA method achieves better performance for the longer batch lengths and longer maximum delays. This is probably because the PNAS version uses IH to compute plans for groups larger than four (see the supplemental material of Alonso-Mora et al. (2017)). This will negatively affect solution quality for easier instances. However, for harder instances, this



**Figure B15.** Computational efficiency analysis of the VGA scenarios during the peak time. In the top figure, we show the evolution of the number of active requests over time. We can see that after the warm-up time, the number of active requests in the system is stable, only slowly decreasing. The middle figure displays the maximum group size that was computed in each batch. The bottom figure demonstrates how the computational time, consisting of the group generation time and the ILP solver time, change during the simulation.

approach may be beneficial compared to our strategy because it allows forming larger groups with potentially suboptimal plans. This observation suggests that a resource-constrained VGA method should use a heuristic to compute larger groups, but the threshold for using this heuristic should be determined by the remaining computational time.

### 3.6. Computational time analysis of the VGA method

Finally, we inspect the computational requirements of the VGA method. In Figure B15, we show the evolution of the number of active requests (top), maximum computed group size (middle), and computational time for group generation and group-vehicle assignment process (bottom) during the peak scenarios, including the warm-up period. Looking at the maximum group size, we see that the 60 ms limit for the group generation results in groups of the maximum size of 5–7 in most batches, while in the optimal scenario, the maximum group size has high variance and goes up to 11.

When we compare the maximum group size with the computation times, we can obtain other valuable insights: (a) the group generation time is strongly

dependent on the maximum group size, and thus it has low variance in the limited scenario and high variation in the optimal scenario, (b) the solver time does not depend on maximum group generation time much, and it is highly variable in both limited and optimal variant, and (c) the group generation time dominates in both scenarios. These findings suggest that further performance optimization of the group generation process may lead to a more favorable tradeoff between the solution cost and computational time.

#### 4. Conclusion

Urban MoD systems represent a promising alternative to private car transport that can reduce the number of vehicles by employing massive vehicle sharing. To further improve the efficiency of an MoD system, the system operator can implement large-scale ridesharing, where multiple passengers are transported in one vehicle simultaneously. Ridesharing can increase vehicle occupancy and reduce the total distance driven in the system, but finding the optimal assignment of passengers to vehicles is a hard combinatorial problem. Traditional exact algorithms for vehicle routing are only applicable to the instances that are orders of magnitude smaller than instances occurring in the metropolitan-scale MoD systems. Therefore simpler heuristic methods for ridesharing are often employed. Recently, the Vehicle-Group Assignment (VGA) has been shown to be capable of solving ridesharing problems with up to 500 vehicles and requests optimally.

In this work, we implemented algorithmic improvements that allowed us to successfully apply the VGA method to a metropolitan-scale MoD system. In contrast to previous studies that sacrifice either scale or optimality, we can regularly compute optimal assignments of more than 21,000 active requests to over 10,000 vehicles. Also, we study the tradeoff between the MoD system efficiency and computational performance for several other passenger-vehicle assignment methods. Specifically, we compared six different scenarios: (1) the "status quo" system with private vehicles, (2) MoD system without ridesharing, (3) MoD system with ridesharing using IH, (4) MoD system with ridesharing using optimal assignments computed by the VGA method, (5, 6) MoD systems with ridesharing that use two resource-limited versions of the VGA method. For all six scenarios, we measured operation cost (total vehicle distance driven), service quality (average delay), fleet size, and congestion levels. Also, we measured the computational time for ridesharing

methods, and in the case of the VGA method, we performed an analysis of the contribution of different subproblems to the overall computational time.

The results confirmed that ridesharing dramatically increases the efficiency of an MoD system: by employing the VGA method, we reduced the total distance driven in the system by more than 57% compared to the present state. Moreover, we demonstrated that the optimal ridesharing assignments are significantly more efficient than assignments computed by the heuristic approach. Our results show that by using the optimal method instead of the IH, we can reduce the total distance traveled by more than 20% while simultaneously reducing the average passenger delay by 5%. Finally, our resource-constrained VGA method provides more than 16% travel distance saving over IH while reducing the computational time by almost 90%. Besides the expected conclusion that ridesharing yields significant savings, these results identify and quantify the optimality gap between a previously proposed resource-constrained version of the VGA method and the optimal solution, and also between a resource constrained VGA method and an IH-based ridesharing method. Our sensitivity analysis provided insights into the limits of the VGA method. The method is capable of finding optimal assignments given that vehicle capacity is no more than 5–10 passengers and the maximum allowed delay is no more than 4–7 minutes, depending on the intensity and structure of the demand. However, for scenarios using higher-capacity vehicles or with more permissive delay constraints, the VGA algorithm can no longer certify the optimality of the computed ridesharing assignments. We believe that all these findings can help future researchers and practitioners to understand the tradeoffs between different MoD system operating policies. Moreover, the optimality gaps provide insights into maximum efficiency gain that one can hope to achieve by developing new heuristic solutions.

In future work, we plan to include more advanced metaheuristics in the comparison. Also, we plan to consider a more general model of a mobility-on-demand system, where travelers could transfer between different vehicles, some of them potentially being fixed-route high-capacity vehicles such as buses or trains. Finally, we plan to investigate the process of MoD system design, including fleet-sizing, fleet composition, and MoD operation from a multi-objective perspective, studying tradeoffs between capital cost, operation cost, and service quality.

## Notes

1. The number of vehicles is smaller than the number of trips. This is possible because even without ridesharing, one vehicle can serve more travel requests sequentially.
2. In practice, we can use lot fewer vehicles, especially for the ridesharing scenarios. However, since the fleet-sizing problem is not the focus of this article, we used this fleet-sizing method to ensure that the size of the fleet is not the limiting factor. The number of vehicles that were actually used in each experiment is in our experimental results.
3. <http://www.gurobi.com/>
4. <https://github.com/aicenter/agentpolis>
5. <https://www.openstreetmap.org/>
6. <https://movement.uber.com/>

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work was supported by the Czech Science Foundation under Grant No. 18-23623S; AMS Institute, and OP VVV MEYS funded project under Grant No. CZ.02.1.01/0.0/0.0/16\_019/0000765 “Research Center for Informatics.” Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the program “Projects of Large Infrastructure for Research, Development, and Innovations” (LM2010005), is greatly appreciated.

## ORCID

David Fiedler  <http://orcid.org/0000-0001-5374-1089>

Javier Alonso-Mora  <http://orcid.org/0000-0003-0058-570X>

Michal Pěchouček  <http://orcid.org/0000-0002-2582-6795>

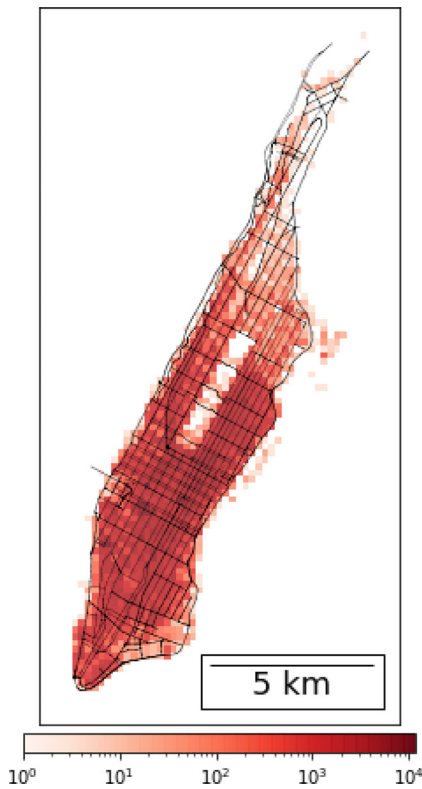
## References

- Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). Ondemand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3), 462–467. <https://doi.org/10.1073/pnas.1611675114>
- Bischoff, J., & Maciejewski, M. (2016). Simulation of city-wide replacement of private cars with autonomous taxis in Berlin. *Procedia Computer Science*, 83, 237–244. <https://doi.org/10.1016/j.procs.2016.04.121>
- Bischoff, J., Maciejewski, M., & Nagel, K. (2017). City-wide shared taxis: A simulation study in Berlin. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 275–280. <https://doi.org/10.1109/ITSC.2017.8317926>
- Campbell, A., & Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems.

- Transportation Science*, 38(3), 369–378. <https://doi.org/10.1287/trsc.1030.0046>
- Čáp, M., & Alonso-Mora, J. (2018). *Multi-objective analysis of ridesharing in automated mobility-on-demand*. Science and Systems XIV.
- Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1), 29–46. <https://doi.org/10.1007/s10479-007-0170-8>
- Drchal, J., Čertický, M., & Jakob, M. (2015). Data driven validation framework for multi-agent activity-based models. International Workshop on Multi-Agent Systems and Agent-Based Simulation, 55–67.
- Drchal, J., Čertický, M., & Jakob, M. (2016). VALFRAM: Validation framework for activity-based models. *Journal of Artificial Societies and Social Simulation*, 19(3), 1–5. <https://doi.org/10.18564/jasss.3127>
- Drchal, J., Čertický, M., & Jakob, M. (2019). Data-driven activity scheduler for agent based mobility models. *Transportation Research Part C: Emerging Technologies*, 98, 370–390. <https://doi.org/10.1016/j.trc.2018.12.002>
- Fiedler, D., Čáp, M., & Čertický, M. (2017). Impact of mobility-on-demand on traffic congestion: Simulation-based study. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 1–6. <https://doi.org/10.1109/ITSC.2017.8317830>
- Fiedler, D., Čertický, M., Alonso-Mora, J., & Čáp, M. (2018). *The impact of ridesharing in mobility-on-demand systems: Simulation case study in Prague*. 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 1173–1178. <https://doi.org/10.1109/ITSC.2018.8569451>
- Fielbaum, A. (2021). Optimizing a vehicle’s route in an on-demand ridesharing system in which users might walk. *Journal of Intelligent Transportation Systems*, 26, 1–20.
- Fielbaum, A., Bai, X., & Alonso-Mora, J. (2021). On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C: Emerging Technologies*, 126, 103061. <https://doi.org/10.1016/j.trc.2021.103061>
- Hensher, D. A., & Button, K. J. (2007). *Handbook of transport modelling*. Emerald Group Publishing Limited.
- Ho, S. C., Szeto, W. Y., Kuo, Y.-H., Leung, J. M. Y., Petering, M., & Tou, T. W. H. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111, 395–421. <https://doi.org/10.1016/j.trb.2018.02.001>
- Jung, J., Jayakrishnan, R., & Young Park, J. (2016). Dynamic shared-taxi dispatch algorithm with hybrid simulated annealing. *Computer-Aided Civil and Infrastructure Engineering*, 31(4), 275–291. <https://doi.org/10.1111/mice.12157>
- Kalina, P., Vokřínek, J., & Mařík, V. (2015). Agents toward vehicle routing problem with time windows. *Journal of Intelligent Transportation Systems*, 19(1), 3–17. <https://doi.org/10.1080/15472450.2014.889953>
- Li, M., Di, X., Liu, H. X., & Huang, H.-J. (2019). A restricted path-based ridesharing user equilibrium. *Journal of Intelligent Transportation Systems*, 24, 1–21.
- Ma, J., Xu, M., Meng, Q., & Cheng, L. (2020). Ridesharing user equilibrium problem under OD-based surge pricing



- strategy. *Transportation Research Part B: Methodological*, 134, 1–24. <https://doi.org/10.1016/j.trb.2020.02.001>
- Ma, T.-Y., Rasulkhani, S., Chow, J. Y. J., & Klein, S. (2019). A dynamic ridesharing dispatch and idle vehicle repositioning strategy with integrated transit transfers. *Transportation Research Part E: Logistics and Transportation Review*, 128, 417–442. <https://doi.org/10.1016/j.tre.2019.07.002>
- Maciejewski, M., & Bischoff, J. (2018). Congestion effects of autonomous taxi fleets. *Transport*, 33(4), 971–980. <https://doi.org/10.3846/16484142.2017.1347827>
- Masmoudi, M. A., Hosny, M., Braekers, K., & Dammak, A. (2016). Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, 96, 60–80. <https://doi.org/10.1016/j.tre.2016.10.002>
- Masoud, N., & Jayakrishnan, R. (2017). A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. *Transportation Research Part B: Methodological*, 106, 218–236. <https://doi.org/10.1016/j.trb.2017.10.006>
- Miller, J., & How, J. P. (2017). *Predictive positioning and quality of service ridesharing for campus mobility on demand systems*. 2017 IEEE International Conference on Robotics and Automation (ICRA), 1402–1408. <https://doi.org/10.1109/ICRA.2017.7989167>
- Muelas, S., LaTorre, A., & Peña, J.-M. (2013). A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Systems with Applications*, 40(14), 5516–5531. <https://doi.org/10.1016/j.eswa.2013.04.015>
- Muelas, S., LaTorre, A., & Peña, J.-M. (2015). A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, 54, 110–130. <https://doi.org/10.1016/j.trc.2015.02.024>
- NYC Taxi & Limousine Commission. (2016). *2016 TLC Factbook*. NYC Taxi & Limousine Commission.
- NYC Taxi & Limousine Commission. (2018). *2018 Factbook*. NYC Taxi & Limousine Commission.
- Pavone, M., Smith, S. L., Frazzoli, E., & Rus, D. (2012). Robotic load balancing for mobility-on-demand systems. *The International Journal of Robotics Research*, 31(7), 839–854. <https://doi.org/10.1177/0278364912444766>
- Pfeiffer, C., & Schulz, A. (2022). An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization. *Or Spectrum*, 44(1), 87–119. <https://doi.org/10.1007/s00291-021-00656-7>
- Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H., & Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37), 13290–13294. <https://doi.org/10.1073/pnas.1403657111>
- Santos, D. O., & Xavier, E. C. (2013). *Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem*. Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2885–2891.
- Santos, D. O., & Xavier, E. C. (2015). Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, 42(19), 6728–6737. <https://doi.org/10.1016/j.eswa.2015.04.060>
- Schrijver, A. (1986). *Theory of linear and integer programming*. John Wiley & Sons, Inc.
- Spieser, K., Treleven, K., Zhang, R., Frazzoli, E., Morton, D., & Pavone, M. (2014). Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in Singapore. In G. Meyer & S. Beiker (Eds.), *Road vehicle automation* (pp. 229–245). Springer International Publishing.
- Shaheen, S., & Cohen, A. (2020). Similarities and differences of mobility on demand (MOD) and mobility as a service (MaaS) | Transportation Sustainability Research Center. *ITE Journal*, 90(6), 29–35.
- Tadaki, S.-i., Kikuchi, M., Fukui, M., Nakayama, A., Nishinari, K., Shibata, A., Sugiyama, Y., Yosida, T., & Yukawa, S. M. (2015). Critical density of experimental traffic jam. In M. Chraïbi, M. Boltes, A. Schadschneider, & A. Seyfried (Eds.), *Traffic and granular flow '13* (pp. 505–511). Springer International Publishing.
- Tamannaï, M., & Irandoost, I. (2019). Carpooling problem: A new mathematical model, branch-and-bound, and heuristic beam search algorithm. *Journal of Intelligent Transportation Systems*, 23(3), 203–215. <https://doi.org/10.1080/15472450.2018.1484739>
- Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications* (2nd ed.). SIAM.
- van Engelen, M., Cats, O., Post, H., & Aardal, K. (2018). Enhancing flexible transport services with demand-anticipatory insertion heuristics. *Transportation Research Part E: Logistics and Transportation Review*, 110, 110–121. <https://doi.org/10.1016/j.tre.2017.12.015>
- Venkatraman, P., & Levin, M. W. (2019). A congestion-aware Tabu search heuristic to solve the shared autonomous vehicle routing problem. *Journal of Intelligent Transportation Systems*, 25, 1–13.
- Wallar, A., Alonso-Mora, J., & Rus, D. (2019). *Optimizing vehicle distributions and fleet sizes for shared mobility-on-demand*. 2019 International Conference on Robotics and Automation (ICRA), 3853–3859. <https://doi.org/10.1109/ICRA.2019.8793685>
- Yan, C.-Y., Hu, M.-B., Jiang, R., Long, J., Chen, J.-Y., & Liu, H.-X. (2019). Stochastic ridesharing user equilibrium in transport networks. *Networks and Spatial Economics*, 19(4), 1007–1030. <https://doi.org/10.1007/s11067-019-9442-5>
- Zhan, X., Szeto, W. Y., & X. M. Chen. (2022). A simulation-optimization framework for a dynamic electric ride-hailing sharing problem with a novel charging strategy. *Transportation Research Part E: Logistics and Transportation Review*, 159, 102615. <https://doi.org/10.1016/j.tre.2022.102615>
- Zhan, X., Szeto, W. Y., Shui, C. S., & Chen, X. (2021). A modified artificial bee colony algorithm for the dynamic ride-hailing sharing problem. *Transportation Research Part E: Logistics and Transportation Review*, 150, 102124. <https://doi.org/10.1016/j.tre.2020.102124>

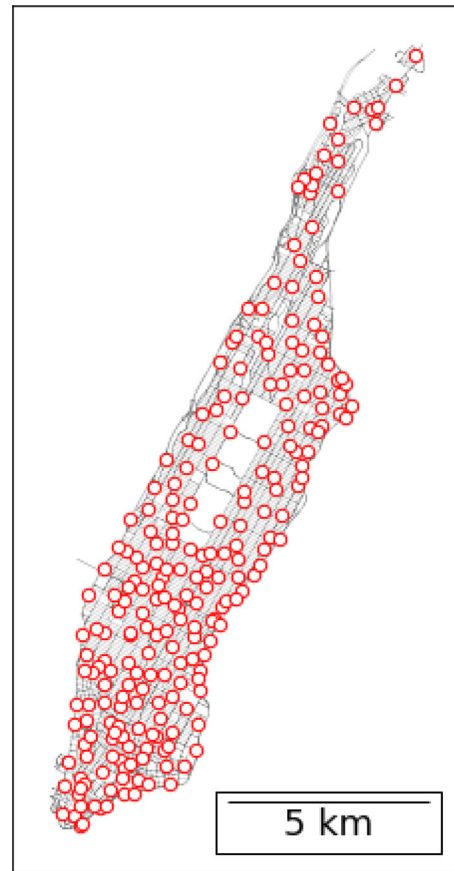


**Figure B16.** Manhattan taxi trip requests on Friday, May 10, 2013, between 19:00 and 20:00. The start positions of all vehicle trips are discretized to squares of 200 square meters. Darker color translates to higher demand, and the color bar has a logarithmic scale.

## Appendix A: Manhattan experiment

In order to demonstrate general applicability of our method and to allow for easier comparison with previous work, we repeat our experiment in Manhattan using a publicly available dataset of taxi trips as transportation demand. Specifically, we use the same demand and road network as used by Alonso-Mora et al. (2017). Identically to our Prague experiment, we simulated the system for one hour with a 30-minute warm-up period. While Alonso-Mora et al. (2017) run the simulation for one week worth of data, here, for simplicity, we selected the day and hour with the largest number of requests, which was Friday, May 10, 2013, between 19:00 and 20:00. There are 137,202 travel request in the selected period, Figure B16 shows the spatial structure of the demand.

Like Alonso-Mora et al. (2017), we use travel speeds along individual road segments derived from historical data, but instead of computing the speeds from the travel demand, we use the speeds from the Uber Movement<sup>6</sup> open data project. Other than that, we followed the methodology described in the main part of this article. That is, we assume a station-based model and perform rebalancing, fleet sizing, and passenger-vehicle matching as described in Sec. 2. Because the historical speeds from Uber Movement dataset are on average approximately half of the posted speed and there are a lot of one-way streets on Manhattan, we need 236 stations to provide the required quality of service, even though Manhattan is about five times smaller than Prague. Figure B17 shows the locations of stations.



**Figure B17.** MoD stations on Manhattan. Each red circle represents a single MoD system station.

On Manhattan, we evaluated five of the six scenarios tested in the Prague case study. We do not evaluate the present state scenario, as the Manhattan dataset represents taxi trips, and therefore, the scenario with MoD system without ridesharing is, in fact, also the “present state” scenario. In Table B5, we can see the results of the same set of experiments as we performed in the Prague case study. Our optimal implementation of the VGA method was able to compute the optimal assignments while the average computational time for a 30 seconds batch was less than 7 seconds. This is in contrast to results reported in Alonso-Mora et al. (2017) that were not computed to optimality and required more than 21 seconds to compute the most similar configuration ( $q_{\max} = 5$  minutes, vehicle capacity of four passengers, 3000 vehicles). This may be because the algorithm by Alonso-Mora et al. (2017) was developed and optimized to allow evaluation of scenarios with even larger delays of 7 minutes and with vehicle capacities of up to 10 passengers; such configurations result in an exponentially larger number of potential passenger-vehicle assignments and consequently, cannot be computed to optimality even with our performance-optimized VGA method.

Because the Manhattan experiment is less complex compared to the Prague experiment, we can observe a similar effect as in the Prague off-peak experiment: the VGA limited method computes only slightly worse solutions than the optimal method, and also the computational times are



**Table B5.** Main results from the Manhattan scenarios during the peak (19:00–20:00) with a maximum passenger delay of 4 minutes.

	No Ridesh.	IH	VGA	VGA lim	VGA PNAS*
Optimal	–	no	yes	no	no
Total veh. dist. (km)	868,899	362,387	334,195	334,737	377,563 (344,057)
Avg. delay (s)	109	117	109	109	83 (110)
Avg. density (veh./km)	0.0183	0.0085	0.008	0.008	0.0089 (0.0082)
Congested seg.	220	9	9	8	14 (10)
Heavily loaded seg.	692	129	117	112	152 (115)
Used vehicles	46,186	20,272	19,714	19,712	22,545 (20,293)
Avg. comp. time (ms)	918	57	6646	6650	24,717 (7170)

Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with a density above 50% of the critical density. For the VGA PNAS method, we also tested a version that does not limit the vehicles considered for each request to 30 nearest vehicles (in parentheses).

**Table B6.** Main results from the Manhattan scenarios during the peak (19:00–20:00) with a maximum passenger delay of 7 minutes and vehicle capacity of 10 persons per vehicle.

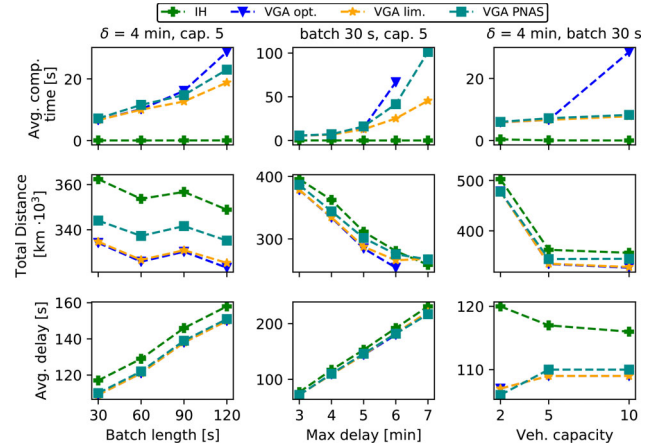
	IH	VGA lim	VGA PNAS*
Optimal	no	no	no
Total veh. dist. (km)	233,859	275,028	267,471
Avg. delay (s)	227	224	217
Avg. density (veh./km)	0.006	0.0067	0.0066
Congested seg.	0	1	0
Heavily loaded seg.	24	53	48
Used vehicles	13,319	16,517	16,025
Avg. comp. time (ms)	25	57,554	139,811

Congested segments are segments on which traffic density is above critical density, and heavily loaded segments are segments with a density above 50% of the critical density. For the VGA PNAS method, we used the version that does *not* limit the vehicles considered for each request to 30 nearest vehicles.

similar. This is because the time limits of the VGA limited method were not reached in the majority of iterations.

Our re-implementation of the PNAS method gives a rather surprising result: the performance metrics are worse than the IH while using more computational time than the optimal method. We investigated this surprising result and found out that the cause is one of the heuristics that limits the number of vehicles considered for assignment to a particular request to 30 nearest vehicles. This heuristic can limit the exploration so much that the solution can be worse than the IH solution. Moreover, for less complex scenarios, the time needed to compute the 30 nearest vehicles can dominate the total computational time, as it happened in our case, probably because this heuristic was not optimized. This observation suggests that in order to achieve acceptable performance, one may need to vary the parameters of heuristics based on the complexity of the problem instance at hand. We also performed the experiment using the VGA PNAS method with this heuristic turned off. For results, see numbers in parentheses in the last column of result tables (Tables B5 and B6).

Table B6 shows another set of results of experiments with  $q_{\max} = 7$  minutes and the capacity of 10 persons per vehicle, which corresponds to the most complex configuration in Alonso-Mora et al. (2017). In this experiment set, we only evaluated the three sub-optimal ridesharing methods to see how they behave under such parametrization. Interestingly, for this scenario, the IH achieves the best performance: The IH finds plans with a total traveled distance that is 12% smaller than plans found by both sub-optimal versions of the VGA method using only a fraction of

**Figure B18.** Sensitivity analysis: Manhattan. Each column represents one experiment set, and inside each column, each value on the x-axis represents one experiment. Each row displays a single measured quantity. The optimal VGA method is only computed for the maximum delay of up to 6 minutes. For the maximum delay of 7 minutes, the optimal ridesharing assignment cannot be computed within 24 hours run time limit.

computational resources. This experiment demonstrates the limit of applicability of the VGA method for routing in large-scale MoD systems. The relaxed time windows and increased vehicle capacity increase the number of feasible groups and the maximum group size to a level that cannot be solved by the ILP solver and the single-vehicle solver, respectively, in practical time. Consequently, the VGA algorithm is unable to return an optimal solution to such instances.

Finally, we performed a sensitivity analysis for the Manhattan case study: the results are reported in Figure B18. It tells a similar story as the sensitivity analysis for the Prague case study (Sec. 3). Some of the previously discussed phenomena are even more apparent in the Manhattan sensitivity analysis. We can see that the computational requirements grow with the maximum delay not only for the optimal VGA method but also for the resource-constrained VGA methods. Also, we can clearly see that the efficiency (total distance driven) gap between IH and the constrained VGA methods is shrinking for larger maximum delays. For the maximum delay of 7 minutes, the IH method starts to outperform both resource constrained VGA methods.

## Appendix B: VGA optimizations

The main objective of this article is to quantify the performance gap between optimal ridesharing assignments and assignments computed using heuristic solutions. However, a naive implementation of the VGA algorithm would require prohibitively long computation time and an extreme amount of memory to compute an optimal solution. In order to arrive at the optimal solutions in a manageable time, we had to implement several performance optimizations. Without these optimizations, the VGA method would need several hours to compute an optimal ridesharing assignment for each 30-second-long batch.

To reduce the number of vehicles considered in request-vehicle matching, we leverage the specific properties of the station-based MoD system and modify the VGA method accordingly. We reduce the number of vehicles for which the groups are generated as follows: First, we observe that we need at most as many vehicles as the number of waiting requests since, in the worst case, each request can be transported in a dedicated vehicle from the nearest station. Second, we exploit symmetries in the solution space. We observe that vehicles parked in a station can be arbitrarily relabeled without any effect on the solution quality. Therefore, instead of computing feasible groups for each vehicle parked in a station, we generate feasible groups for only one vehicle from that station, representing any other vehicle currently parked in the station. Consequently, in the assignment ILP, we can relax Constraint (7) corresponding to this representative vehicle to allow assigning as many vehicle plans as there are vehicles parked in the station:

$$\sum_{g=1}^{|\Gamma_v|} \xi_v^g \leq |V_s| \quad \forall s \in S. \quad (\text{B1})$$

In this modified version of Constraint (7),  $V_s$  is the set of vehicles parked in station  $s$ , and  $S$  is the set of all stations.

We paid special attention to an efficient implementation of the function  $f$  that is used to determine the feasibility of a newly formed group and to compute the optimal route for the group. This function solves a single-vehicle DARP by searching through all feasible permutations of travel schedules. Most of the time during group generation is spent inside this function. We achieved significant performance gains by implementing the algorithm in a way that constructs permutations “in place” and avoids memory allocation during the search process. Also, we implemented a look-ahead procedure that is triggered each time a pickup or drop-off location is added to extend a partial plan. In this look-ahead, we verify that the maximum allowed time for each pickup and drop-off location that are still waiting to be added to the plan is higher than the time of the most recently added order to the partial plan; if the above does not hold, we can safely discard the partial plan as infeasible.

Finally, we parallelized the group generation process so that feasible groups are computed in a separate thread for each vehicle. An even better approach would be to parallelize the function  $f$  because that way, we can distribute the work among threads even if there is only a small number of vehicles with a larger number of vehicle groups. However, we leave this optimization for future work.