Large-scale Online Ridesharing: The Effect of Assignment Optimality on System Performance

David Fiedler[a], Michal Čertický[a], Javier Alonso-Mora[b], Michal Pěchouček[a] and Michal Čáp[a]

[a]Department of Computer Science, Faculty of Electrical Engineering, CTU in Prague, Czech Republic; [b]Department of Cognitive Robotics, 3ME, TU Delft, Netherlands

ARTICLE HISTORY
Compiled May 4, 2023

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.

KEYWORDS
Vehicle routing, Traffic control, Simulation, Ridesharing, Mobility-on-Demand

1. Introduction

In densely-populated cities, private cars are considered to be an unsustainable mode of transportation. Typically, the parking capacity and the 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 space, and air pollution.

One of the proposed remedies to these problems is a large-scale deployment of metropolitan mobility-on-demand systems (MoD) that would serve as an alternative to private transportation. An MoD system consists of a fleet of shared vehicles that jointly serve the travel requests of the system's users. For each incoming travel request,

CONTACT David Fiedler. Email: david.fiedler@agents.fel.cvut.cz

the MoD system dispatches a vehicle to transport the passenger to the desired location. Examples of MoD systems include the existing transportation companies such as Uber or Lyft, as well as the future systems of autonomous self-driving cars being developed by companies such as Waymo.

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. 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.

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; Tamannaei & 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 existing exact solution methods focus on small-scale instances with tens of vehicles and requests (Cordeau & Laporte, 2007; Ho et al., 2018) making them unsuitable for large-scale fleets in urban mobility-on-demand systems consisting of thousands of vehicles. For example, in New York City (NYC), there are almost 100 000 active taxis per hour during peak traffic (NYC Taxi & Limousine Commission, 2018).

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 (Ma et al., 2019), or as an initial solution generator for metaheuristic methods (Muelas et al., 2013). The meta-heuristic 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), and suffer from scalability issues when applied to large-scale DARPs. A popular meta-heuristic is simulated annealing by Jung et al. (2015). One of the largest scenarios solved using this method contains 600 operating vehicles. Other popular meta-heuristics include Greedy Randomized Adaptive Search Procedure (GRASP) by Santos and Xavier (2013), and Adaptive Large Neighborhood Search (ALNS) (Masmoudi et al., 2016) that, however, are limited to even smaller problem instances. 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.

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 trade-offs between the quality of service and the operation cost inherent in ridesharing.

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). 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, they computed the routes for groups with more than four passengers heuristically, and they terminated the assignment optimization process after 15 seconds. In this work, we identified and solved several algorithmic bottlenecks, and consequently, we were able to obtain optimal solutions. As an example, we discovered that the size of the optimization problem can be significantly reduced by an equivalent reformulation that replaces all binary decision variables for vehicles parked in a station with one non-binary decision variable.

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: We quantify the potential to reduce the fleet size, the amount of vehicle distance traveled, the quality of the service, and the traffic load that can be achieved by employing different ridesharing strategies. Specifically, we compare the above metrics in five 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, and e) an MoD system with ridesharing solved by a resource-limited VGA method. 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 %.

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. 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 of Prague and Central Bohemian Region (Čertický et al., 2015). 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 system performance.

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 workday 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 machine learning methods (neural networks, decision trees, random forests, etc.) and trained using various real-world data sets, including census data, travel diaries, and other transportation-related surveys. 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., 2016; Drchal et al., 2015). The model generates over three million trips by all modes of transport in one 24-hour scenario, out of which there are roughly one million trips by private vehicles (Figure 1). In this work, we select only the trips realized by private vehicles in two representative time intervals: the *peak* dataset includes trips that start between 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.

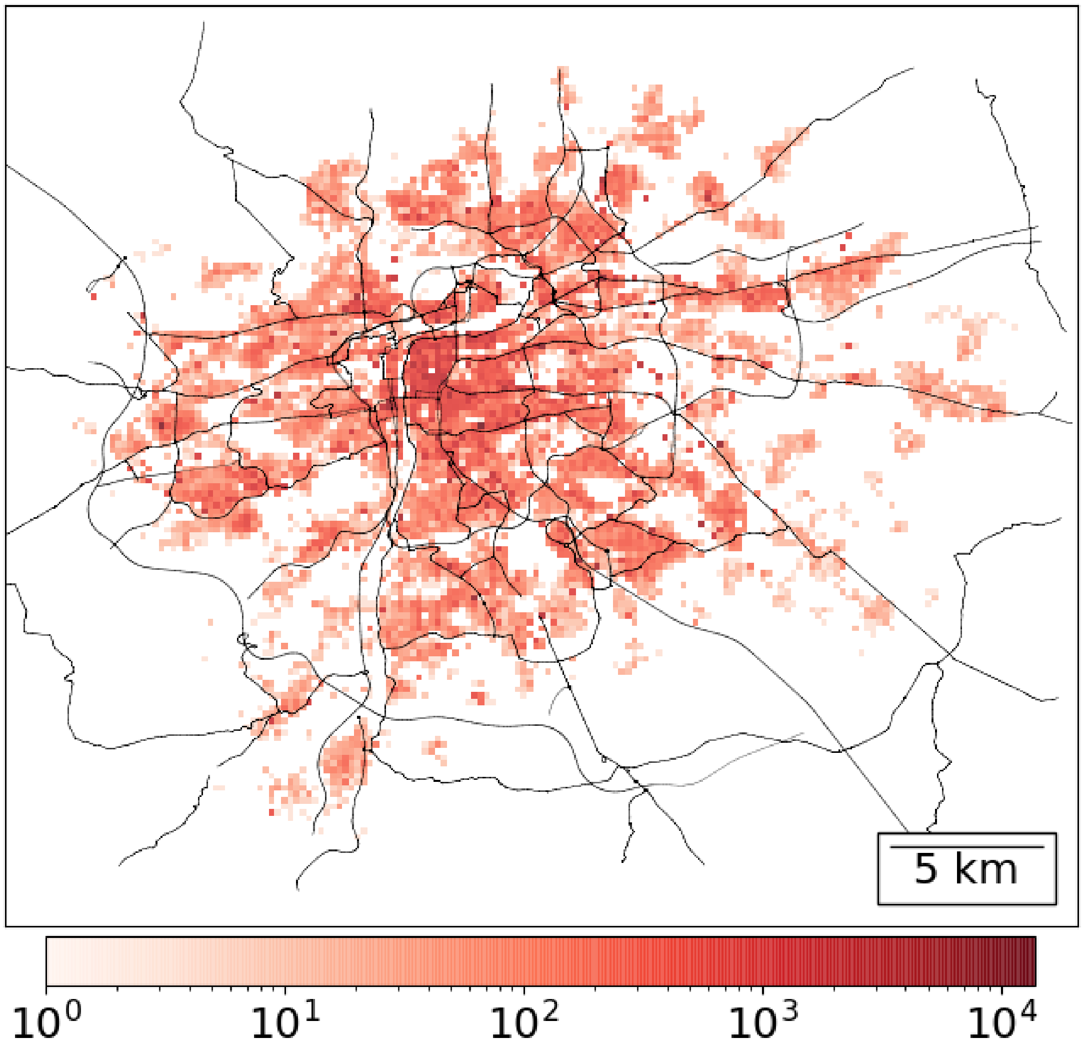


Figure 1.: Demand for personal vehicle traffic in Prague. The start positions of all vehicle trips are discretized to squares of 200 square meters. Darker color translates to higher demand, and the color bar has a logarithmic scale.

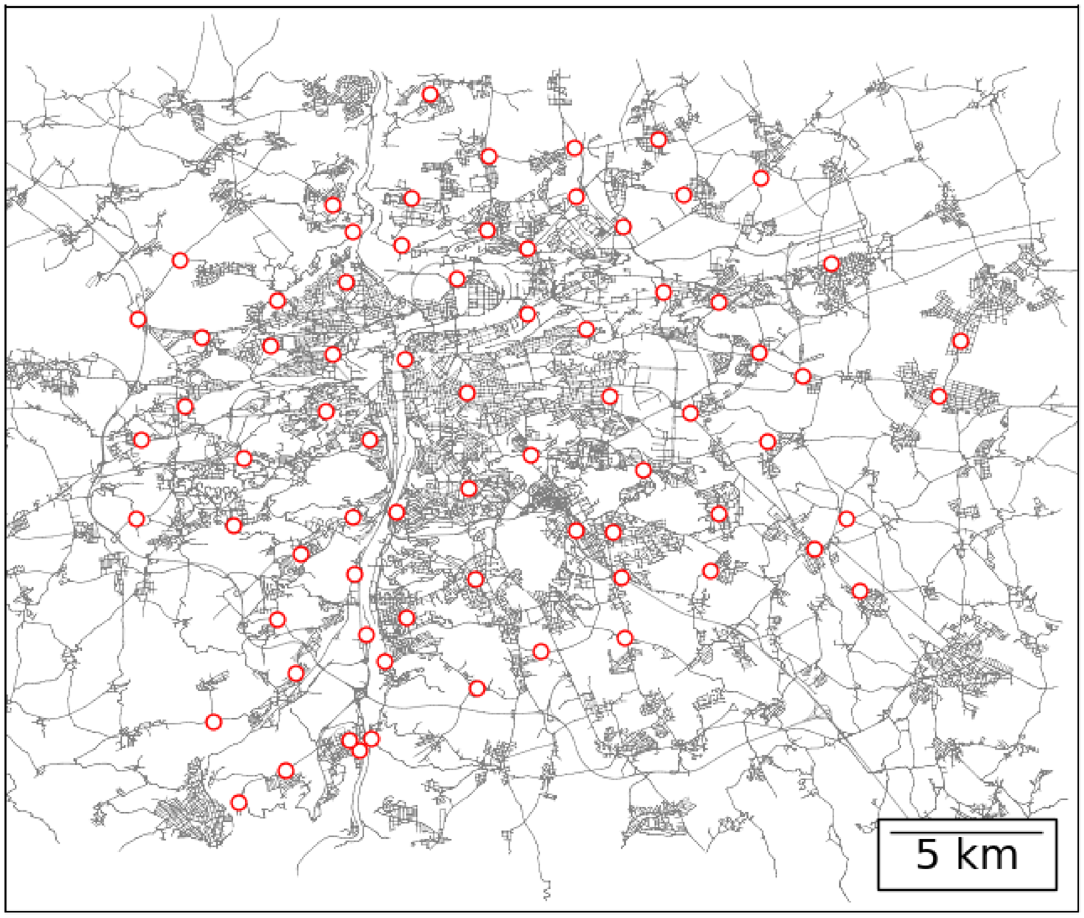


Figure 2.: MoD system stations in the city of Prague. There are 73 stations in total, shown as red circles.

2.2. System model

We adopt a station-based methodology for the design of MoD systems (Wallar et al., 2019). In this methodology, the vehicles make use of *stations* that contain facilities such as temporary parking, refueling/charging, and cleaning. We use 73 stations shown in Figure 2 chosen such that every point on the road network (excluding roads without travel requests such as tunnels, highways, etc.) can be reached from one of the stations within four minutes.

2.2.1. MoD System Design

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. The set of all vehicles will be denoted as $V = 1, \ldots, m$. 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. 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 Section 2.1) without ridesharing, the MoD system requires a total of 68 201 vehicles, and we used the same fleet size also for the off-peak scenarios. Specifically, to determine the number of vehicles 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 in each station until the first vehicle shortage event occurred in the scenario without ridesharing. This procedure guarantees that there is a sufficient number of vehicles to serve all requests from the nearest station.

2.2.2. Problem Formulation

Travel requests are modelled as a sequence $(t_1, o_1, d_1), (t_2, o_2, d_2), \dots$, 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.

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. The travel delay of request r when it is served by vehicle v following plan p is denoted $q_r(p, v)$ and is defined as the difference between the travel time in a shared vehicle and a travel time along the fastest route:

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

Here, t_r^{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. Our goal is to minimize the total operational cost of the system, such that the discomfort of every passenger is bounded by a constant $q_{\max}$. That is, we desire to minimize

$$\sum_v c(p, v)$$

subject to

$$q_r(p, v) \le q_{\max}$$

for each request r, while serving all requests.

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

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      $v^* \leftarrow$ null ;
4      for $v \in V$ do
5          for $i \in 1, \dots, |p_v|$ do
6              for $j \in x+1, \dots, |p_v|+1$ 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 $p^*$
```

one batch, we collect all newly announced requests and execute a planning procedure that computes request-vehicle matching and corresponding vehicle plans.

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 every 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^{new}. Note that the relative ordering of all locations from p_v remains unchanged in the new plan. 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 2: VGA method

```
input  : The current state of vehicles $V$ and the set of waiting requests $D_w$.
1 for $v \in V$ do
2     $\Gamma_v \leftarrow$ generate_groups($v$, $D_w$);
3 $\pi* \leftarrow$ Solve Problem 1 using $\Gamma_1 \dots \Gamma_m$;
4 All vehicles follow the optimal system plan $\pi*$
```

2.3.2. Vehicle Group Assignment Method

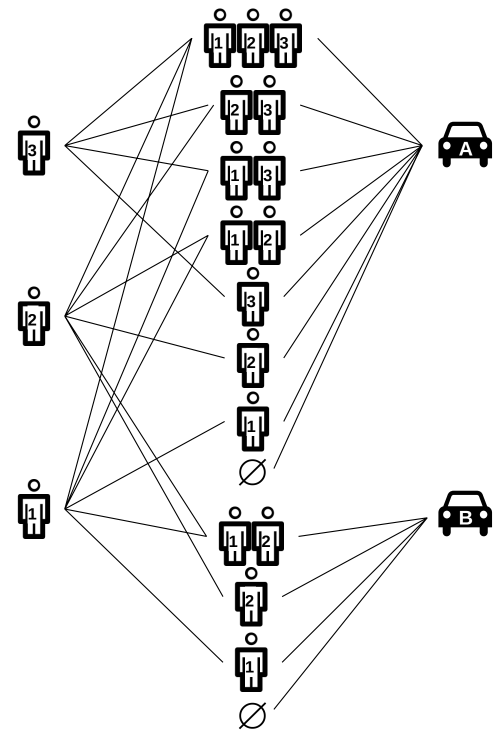
The VGA method relies on the performance improvement coming from conversion of a DARP problem to a variant of assignment problem. In this work, we generalize the formulation by Čáp and Alonso-Mora (2018) to be applicable in an online optimization setting. That is, we reformulated the algorithm to be able to deal with unknown future demand.

The VGA method can be divided into two phases: group generation (Algorithm 3) and vehicle-group assignment (Problem 1). We can see the overall pseudocode in Algorithm 2. Let D_w 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 service quality guarantees using the *group generation* algorithm (Algorithm 3). The second phase uses ILP (Problem 1) to map exactly one group to every vehicle so that every request is serviced and the system plan is optimal. The whole procedure is demonstrated by an example in Figure 3.

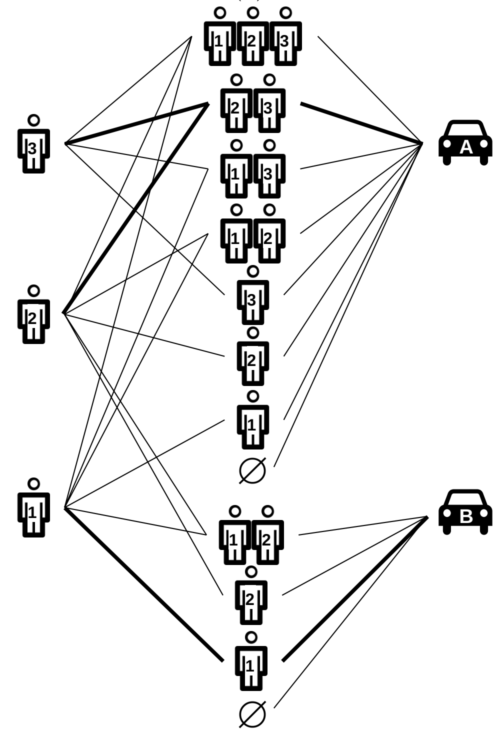
We say that a group R is *feasible* for vehicle v if a feasible plan exists for the vehicle that serves all requests from R and if all requests onboard vehicle v are members of the group. We denote a set of all groups feasible for vehicle v by Γ_v. The key property of feasible groups, observed by Alonso-Mora 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 Γ_v^1. Then, we find larger groups iteratively by combining the feasible groups from the previous iteration with all feasible requests. At the end of this step, we have a set of feasible groups for each vehicle, as it is illustrated in Figure 3a.

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 ξ_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:



(a)



(b)

Figure 3.: Example of the VGA method assigning three passengers to two vehicles. In Figure 3a, we show 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. In Figure 3b, the final assignment between vehicles and groups is shown (bold lines).

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.

```
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      $\Gamma_v^{k+1} \leftarrow \emptyset$;
       /* not check groups repeatedly                        */
9      checked $\leftarrow \emptyset$;
10     forall $R \in \Gamma_v^k, \{r\} \in \Gamma_v^1$ do
11         if $(R \cup \{r\}) \notin$ checked and $\forall R' \subset (R \cup \{r\}), |R'| = k : R' \in \Gamma_v^k$ and
            $f(R \cup \{r\}, v)$ then
12             $\Gamma_v^{k+1} \leftarrow (R \cup \{r\}) \cup \Gamma_v^{k+1}$;
13         checked $\leftarrow$ checked $\cup (R \cup \{r\})$
14     $k \leftarrow k + 1$;
15 if $|R^{\text{init}}| > 0$ then
16     $\Gamma_v^1 \leftarrow \emptyset$;
17 $\Gamma_v \leftarrow \{\emptyset\} \cup \Gamma_v^1 \cup \Gamma_v^2 \cup \cdots \cup \Gamma_v^k$;
```

Problem 1 (Vehicle-group Assignment)

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

subject to

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

$$\sum_{v=1}^{m} \sum_{g=1}^{|\Gamma_v|} \mathbf{1}_{R_g}(r) \xi_v^g = 1 \quad \forall r \in D_w \tag{2}$$

In the objective function, p_v^{g*} denotes the optimal plan for vehicle v to serve group R_g. Constraint 1 states that only one group can be assigned to each vehicle. Constraint 2 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 3b for example assignment). This assignment can be directly translated into vehicle plans that replace vehicle plans from the previous iterations.

2.4. Ridesharing Implementation

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.[1] 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 %. Note that both the vehicle-group assignment and the group generation processes are very complex combinatorial problems. In particular, the group generation requires a lot of computational time because the function $f(R, v)$ needs to be implemented as an exhaustive search to guarantee optimality. In order to demonstrate the anytime property of the VGA method and its ability to achieve different trade-offs between ridesharing efficiency and computational cost, we also tested a more resource-constrained setup of the VGA method with ILP solver runtime limited 30 s (using the `TimeLimit` parameter of the Gurobi solver) and group generation time-limited to 30 ms per vehicle.

We compute passenger-vehicle assignments together with the simulation sequentially, and thus from a simulation perspective, the ridesharing computation is an instantaneous event. In the 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.

Finally, we modified the VGA method to leverage the specific properties of the station-based MoD system. If the groups were generated for all vehicles that are parked in the stations, both the group generation and the vehicle group assignment process would become computationally intractable. Therefore, 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 will 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, we generate only one set of feasible groups that represents feasible groups for any vehicle currently parked in the station. Consequently, in the assignment ILP, we can relax Constraint 1 corresponding to this set of feasible groups to allow selecting as many groups as there are vehicles parked in the station.

2.5. Simulation

In our experiments, we simulated the following five scenarios:

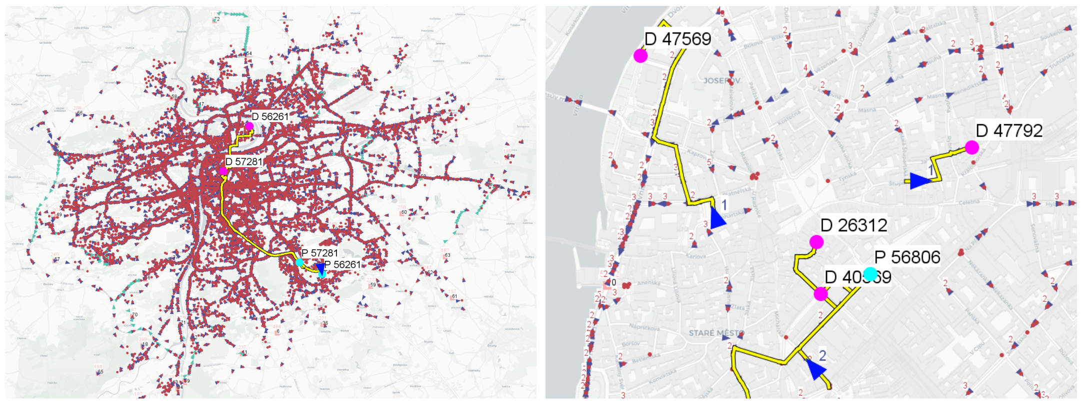
- *Present state*: All the requests are served by private vehicles. The vehicles are parked at the request's start location, i.e., there is no delay. The number of used vehicles is equal to the number of requests, and the total distance traveled is equal to the sum of the shortest paths between the origins of all requests and their destinations.
- *MoD w/o ridesharing*: MoD system without ridesharing, the plans are computed

[1]http://www.gurobi.com/

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 ridesharing 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 30 ms per vehicle and the maximum ILP solver time of 30 s.

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 min 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[2]. 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 4, we show a screenshot of the AgentPolis visualization captured during one of the simulation experiments.



(a) (b)

Video: https://sum.fel.cvut.cz/agentpolis/

Figure 4.: AgentPolis visualization of the simulated traffic in Prague during the traffic peak. Figure 4a (left): the entire city of Prague in the simulation. A more detailed (zoomed in) view can be seen in Figure 4b (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 Figure 4b, 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).

We use an OpenStreetMap[3] road network consisting of 158 674 edges and 63 995 nodes. The speed limit for each road segment was also taken from OpenStreetMap

[2]https://github.com/aicenter/agentpolis
[3]https://www.openstreetmap.org/

	Present	Mobility-on-Demand			
		No Ridesh.	IH	VGA opt.	VGA lim.
Total veh. dist. (km)	**757 984**	**1 002 661**	**537 500**	**430 347**	**463 276**
Avg. delay (s)	-	132	190	180	176
Avg. dens. (veh/km)	0.0078	0.0086	0.0054	0.0046	0.0050
Congested seg.	9	27	2	0	0
Heavily loaded seg.	170	303	35	15	22
Used Vehicles	122 473	33 057	15 553	13 925	14 744
Avg. comp. time (ms)	-	12	10	176 039	33 861

Table 1.: Main results from the considered scenarios during the morning peak (7:00-8:00). 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.

data, and missing entries were generated according to 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 Section 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.

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 1 and 2 summarize the main results of the experiments. As explained in Section 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 almost 107 153 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 572 314 km (57 %) and 327 637 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

		Mobility-on-Demand			
	Present	No Ridesh.	IH	VGA opt.	VGA lim.
Total veh. dist. (km)	**283 415**	**344 382**	**210 879**	**175 591**	**178 154**
Avg. delay (s)	-	131	191	179	179
Avg. density (veh/km)	0.0045	0.0047	0.0034	0.0031	0.0031
Congested seg.	0	0	0	0	0
Heavily loaded seg.	9	11	4	1	1
Used Vehicles	42 633	7670	4641	4748	4877
Avg. comp. time (ms)	-	1	3	5431	4188

Table 2.: Main results from the considered scenarios, off-peak (11:00-12:00). 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.

per one optimization batch in the peak scenario was about 177 s, compared to 10 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 five-fold over the unconstrained version of the VGA method while still reducing the total traveled distance by more than 14 % over the IH. In the off-peak scenario, the limited version of the VGA method performs almost the same as the unconstrained version because the time limits are rarely reached.

3.2. Trade-off 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 1 and 2, we can see that the optimal VGA method saves about 5 % time over the IH in both peak and off-peak experiments. The trade-off between the operating cost (distance traveled) and the service quality (average delay) is depicted in Figure 5.

A more detailed overview of the passenger delays with a delay histogram for the four MoD scenarios in both time windows is in Figure 6. 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.

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 vehicle m^{-1} (Tadaki et al., 2015) as *congested*. Segments with density above

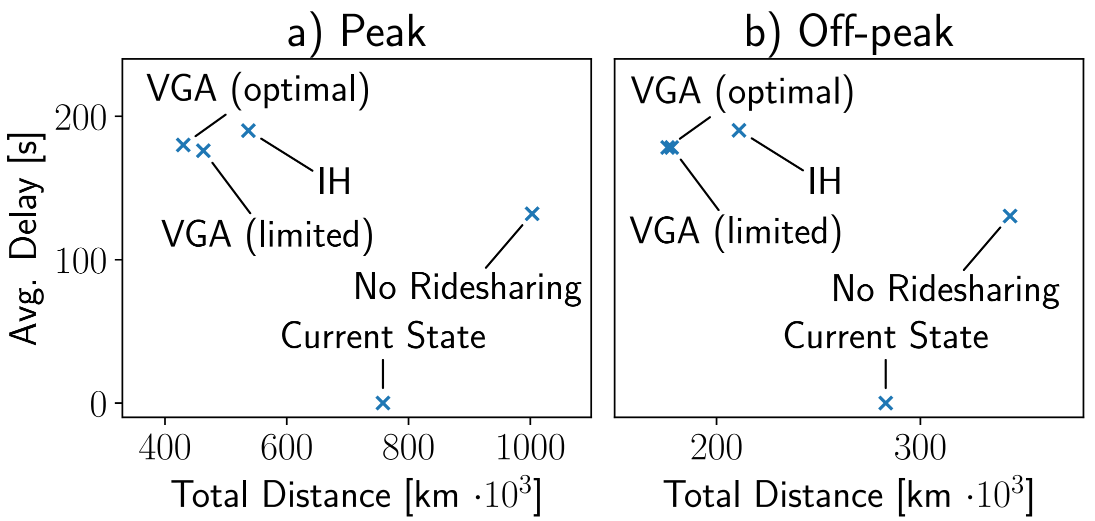


Figure 5.: The trade-off between total distance traveled by all vehicles and the average delay of one passenger trip.

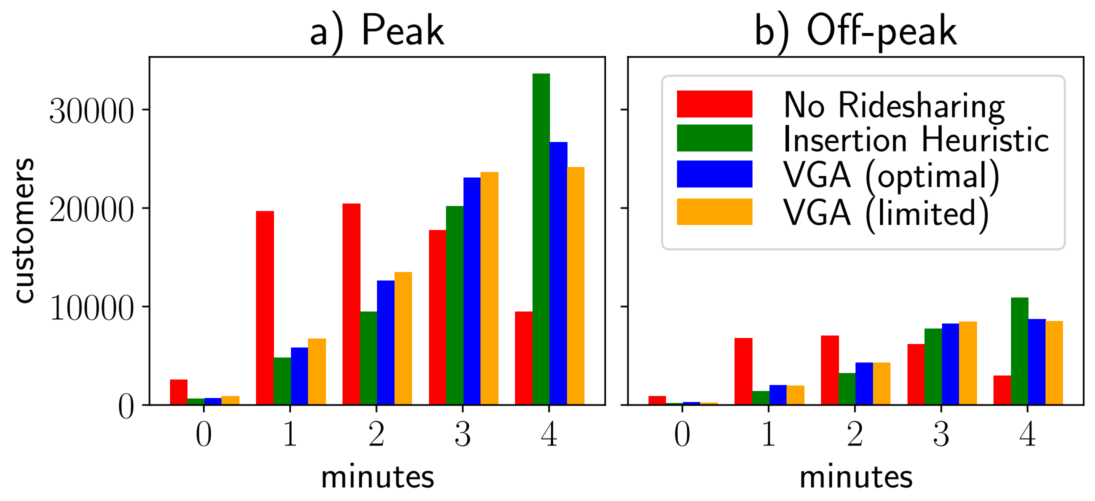


Figure 6.: Histograms of delays. The present state scenario is omitted as the delay is always zero.

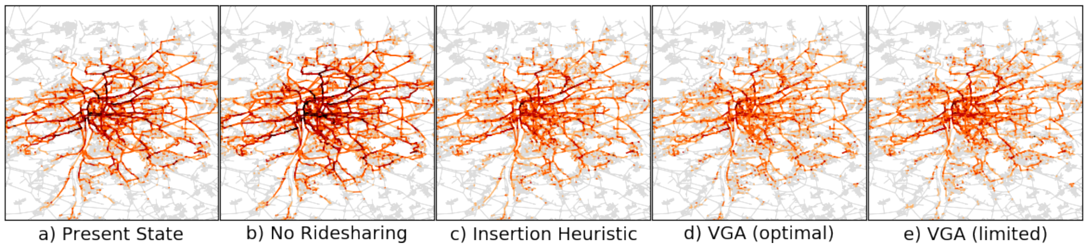


Figure 7.: Traffic density map of the four scenarios during the morning peak. Darker colors signalize higher traffic density. Black color means that the road segment is congested.

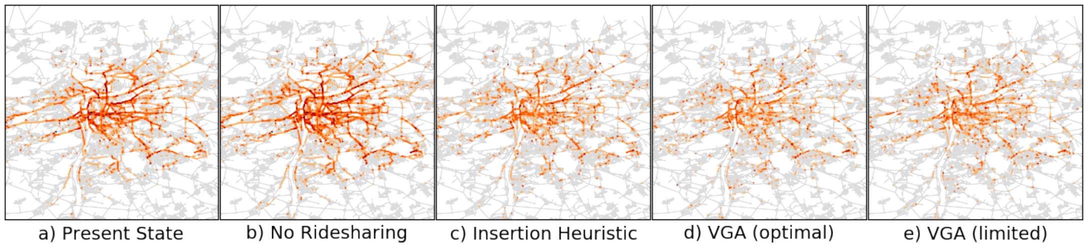


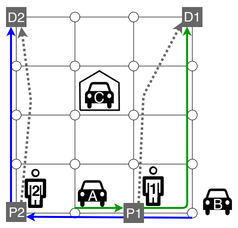
Figure 8.: Traffic density map of the four scenarios during the off-peak time. Darker colors signalize higher traffic density. Black color means that the road segment is congested.

$0.04\,\mathrm{vehicle\,m^{-1}}$ are considered as *heavily loaded*. As you can see in Table 1, in the morning peak, using the optimal VGA method reduces the average traffic density by 14 % over the ridesharing that uses IH, and by 46 % and 41 % 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 7 and 8 depict traffic densities on every road for all five scenarios.

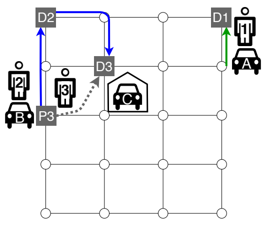
3.4. Fleet Size and Vehicle Occupancy

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 1628 (10 %) fewer 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.

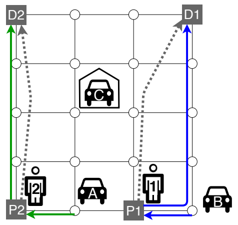
In the off-peak time, however, we registered that the optimal VGA method uses about 2 % more vehicles than IH. By analyzing the simulation output, we found an explanation for this perhaps surprising result. First, counterintuitively, it is possible that a suboptimal vehicle assignment that generates plans with longer total distance can lead to fewer vehicles being used, as it is illustrated in Figure 9. Second, by analyzing the vehicle trips in both IH and VGA scenario, we found that such situations



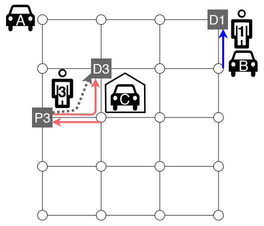
(a) IH Iteration 1



(b) IH Iteration 2



(c) VGA Iteration 1



(d) VGA Iteration 2

Figure 9.: Example of the capital cost paradox. In Figures 9a and 9b, we can see two iterations of the IH. In Figure 9a, 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. In Figure 9b, 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 ((9c) and (9d)) 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.

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: 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 trade-offs associated with different design and control strategies.

Next, we measured vehicle occupancy: Figure 10 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. Computational Time Analysis of the VGA Method

Finally, we analyzed various metrics for the VGA scenarios. In Figure 11, 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

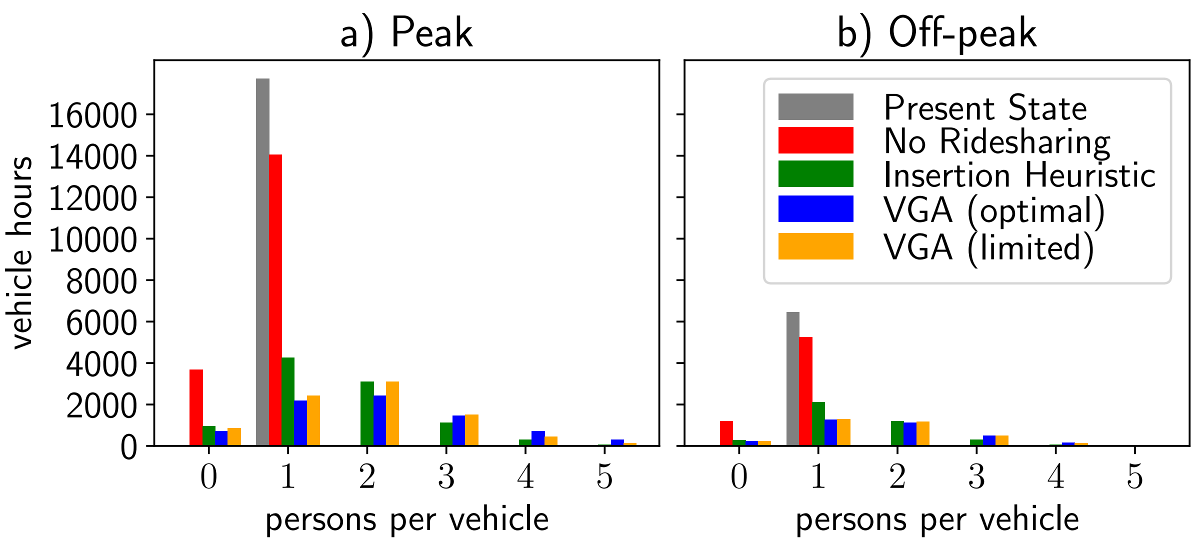


Figure 10.: Occupancy histogram of all five scenarios.

process (bottom) during the peak scenarios, including the warm-up period. Looking at the maximum group size, we see that the 30 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 the optimal scenario. These findings suggest that a VGA method with a limit only on group generation time could achieve an even better trade-off 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 trade-off between the MoD system efficiency and computational performance for several other passenger-vehicle assignment methods. Specifically, we compared five 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, and 5) MoD system with ridesharing that uses a resource-limited version of the VGA method. For all five 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 sub-problems 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 14 % travel distance saving over IH while reducing the computational time by more than 80 %.

We believe that these findings bring valuable insight into the potential of ridesharing in MoD systems and that it can help both researchers and practitioners to understand the trade-offs between different MoD system operating policies. In future work, we plan to include more advanced metaheuristics in the comparison. Also, we plan to investigate the process of MoD system design, including fleet-sizing, fleet composition, and MoD operation from a multi-objective perspective, studying trade-offs between capital cost, operation cost, and service quality.

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.

References

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462–467.

Bischoff, J., & Maciejewski, M. (2016). Simulation of City-wide Replacement of Private Cars with Autonomous Taxis in Berlin. *Procedia Computer Science*, *83*, 237–244.

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.

Campbell, A., & Savelsbergh, M. (2004). Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science*, *38*, 369–378.

Čáp, M., & Alonso-Mora, J. (2018). Multi-Objective Analysis of Ridesharing in Automated Mobility-on-Demand. *Robotics: Science and Systems XIV*.

Čertický, M., Drchal, J., Cuchý, M., & Jakob, M. (2015). Fully agent-based simulation model of multimodal mobility in european cities. *International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 229–236.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, *153*(1), 29–46.

Drchal, J., Čertický, & Jakob, M. (2016). VALFRAM: Validation Framework for Activity-Based Models. *Journal of Artificial Societies and Social Simulation*, *19*(3), 1–5.

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.

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.

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.

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.

Jung, J., Jayakrishnan, R., & Young Park, J. (2015). Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing. *Computer-Aided Civil and Infrastructure Engineering*, *31*.

Kalina, P., Vokřínek, J., & Mařík, V. (2015). Agents Toward Vehicle Routing Problem With Time Windows. *Journal of Intelligent Transportation Systems*, *19*(1), 3–17.

Li, M., Di, X., Liu, H. X., & Huang, H.-J. (2019). A restricted path-based ridesharing user equilibrium. *Journal of Intelligent Transportation Systems*, *0*(0), 1–21.

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.

Maciejewski, M., & Bischoff, J. (2018). Congestion effects of autonomous taxi fleets. *Transport*, *33*(4), 971–980.

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.

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.

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.

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.

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.

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.

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.

Spieser, K., Treleaven, 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.

Tadaki, S.-i., Kikuchi, M., Fukui, M., Nakayama, A., Nishinari, K., Shibata, A., Sugiyama, Y., Yosida, T., & Yukawa, S. (2015, November 1). Critical Density of Experimental Traffic Jam.

Tamannaei, 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.

Toth, P., & Vigo, D. (2014, December 5). *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. 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.

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*, *0*(0), 1–13.

Wallar, A., Alonso-Mora, J., & Rus, D. (2019). Optimizing Vehicle Distributions and Fleet Sizes for Mobility-on-Demand, 7.

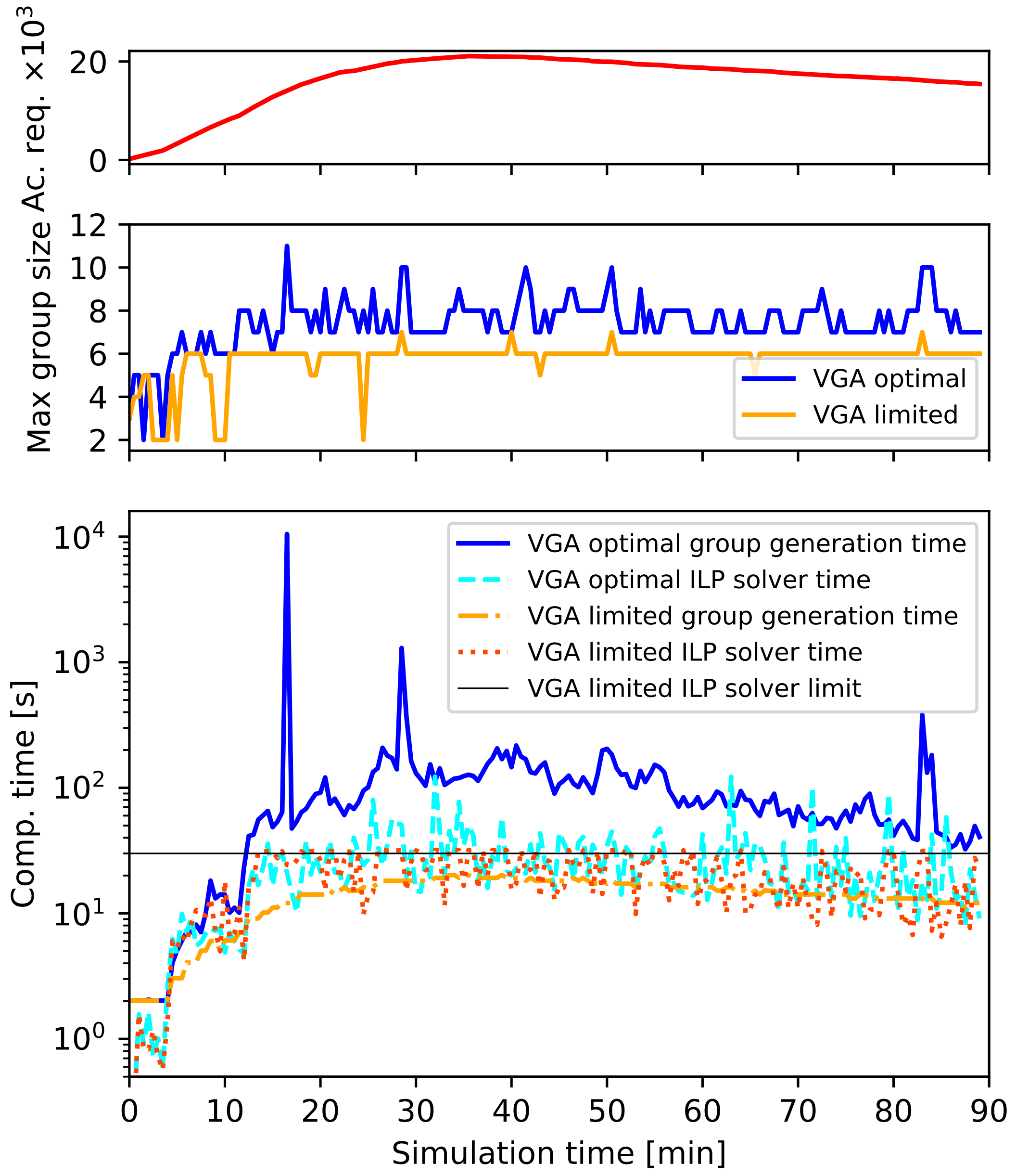


Figure 11.: Computational efficiency analysis of the VGA scenarios during the peak time. In the top figure, we show the evolution of the number of active requests over time. We can see that after the warm-up time, the number of active requests in the system is stable, only slowly decreasing. The middle 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.