Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Scalable Offline Ridesharing Algorithm

Olga Kholkovskaia

Supervisor: Ing. David Fiedler
Field of study: Open Informatics
Subfield: Computer and Information Science
January 2020

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kholkovskaia Olga** Personal ID number: **464308**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Scalable Offline Ridesharing Algorithm

Bachelor's thesis title in Czech:

Škálovatelný algoritmus pro offline ridesharing

Guidelines:

1. Explore the state-of-the-art methods for ridesharing with focus on offline ridesharing.
2. Create a short review of ridesharing algorithms. Categorize the methods (optimal/heuristic, online/offline, ...) and compare them (scalability, efficiency, ...).
3. Design a method for offline ridesharing that can solve metropolitan-scale scenarios.
4. Implement the designed method and prepare it for the integration to other software.
5. Compare the method with at least one other heuristic ridesharing method. Analyze the differences in scalability and ridesharing efficiency.

Bibliography / sources:

[1] D. Fiedler, M. Čertický, J. Alonso-Mora and M. Čáp, "The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 1173-1178.
[2] Jung, Jaeyoung & Jayakrishnan, R & Young Park, Ji. (2015). Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing. Computer-Aided Civil and Infrastructure Engineering. 31. 10.1111/mice.12157.
[3] M. Cap and J. Alonso-Mora, "Multi-Objective Analysis of Ridesharing in Automated Mobility-on-Demand," in Robotics: Science and Systems XIV, 2018.
[4] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," in 2017 IEEE RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 3583–3590.
[5] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing Vehicle Distributions and Fleet Sizes for Mobility-on-Demand," presented at the IEEE International Conference on Robotics and Automation (ICRA), 2019, p. 7.

Name and workplace of bachelor's thesis supervisor:

Ing. David Fiedler, Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **12.06.2019** Deadline for bachelor thesis submission: **07.01.2020**

Assignment valid until: **19.02.2021**

Ing. David Fiedler Supervisor's signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

At the beginning of this work I would like to thank my research supervisors, Ing.David Fiedler for for his for providing guidance and feedback throughout this project, and also Ing. Martin Schaeffer and Ing. Michal Čáp for cooperation and support during my previous semestral project.

V úvodu mé bakalářské práce bych ráda poděkovala vedoucímu prace panu Ing. Davidu Fiedlerovi za vedení při jejím psaní, cenné rady a vstřícný přístup, dále panu Ing. Martinu Schaefferovi a panu Ing. Michalu Čápovi za spolupráci a podporu v předcházejícím semestrálním projektu.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within in accordance with methodical instructions for observing ethical principles in the preparation of university theses.

Prague, 1. January 2020

Prohlašuji, že jsem předloženou práci vypracovala samostatně, a že jsem uvedla veškeré použité informacní zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 1. January 2020

Abstract

This work makes an attempt to apply a combination of known methods designed for large dynamical on-demand routing and scheduling problems to solve static Dial-a-Ride Problem of comparable size. Our main task is to design and implement a solution for offline ridesharing able to handle large city-scale scenarios. The proposed solution is evaluated on real-life data from New York City and compared with Insertion Heuristic, another ridesharing method widely used for large-scale problems.

Keywords: ridesharing, DARP

Supervisor: Ing. David Fiedler
Department of Computer Science,
Czech Technical University in Prague,
Karlovo náměstí 13,
121 35 Prague 2

Abstrakt

Cílem této práce je návrh a realizace škálovatelného algoritmu pro offline ridesharing a jeho porovnání s Insertion Heuristic, metodou široce používanou při řešení velkých dynamických a statických problémů. Navrženo řešení je kombinaci dvou známých metod určených primárně pro online ridesharing. Řešení je experimentálně evaluováno na reálných datech z New Yorku.

Klíčová slova: ridesharing, DARP

Contents

Figures

Tables

Chapter 1

Introduction

Term ridesharing is commonly used nowadays to describe a road transportation system that receives on-demand requests from users and schedules a fleet of vehicles to serve those requests so that individual passengers may share a single vehicle during the trip.
According to the annual INRIX Global Traffic Scorecard, an average driver in New York City loses 133 hours in congestions with cost per driver amounting to $1,859. Almost 70% congestion level during evening rush hours adds 21 extra minutes to a 30-minute trip [INR18; Tom18]. Ridesharing may be beneficial for all the participants and society in general. Shared vehicle implies a reduced cost for individual travellers. And fewer vehicles globally means increased speed, reduced congestions, and air pollution.
The adoption of shared transportation requires effective routing solutions able to coordinate travellers and vehicles [Fur+13]. On a city level, that means scheduling from tens to hundreds of thousands of requests each hour. Neither exact methods nor heuristics designed for offline ridesharing can handle problems of that scale. On the other hand, solutions designed for large-scale transportation problems are meant to work in online mode, which ignores some of the offline constraints and cannot use exact information about future requests. Moreover, being designed to solve parallel, not serial rides, such solutions do not scale well over the long time horizon.
In this work, we use a modification of two known methods designed for large dynamical on-demand routing and scheduling problems to solve a large-scale static offline problem. Different types of offline problems and methods used for real-life dynamic transportation problems are summarized in Chapter 2. The proposed solution is tested and evaluated on real data from New York City and compared with Insertion Heuristic, another ridesharing method widely used for large-scale problems. The solution is presented in Chapter 4. Results of evaluation against Insertion Heuristic as a baseline are reported in Chapter 5.

Chapter 2

State Of the Art

Ridesharing or Dial-a-Ride Problem (DARP) can be viewed as a special case of Pickup-and-Delivery Problem (PDP) or Vehicle Routing Problem (VRP) [CL03b]. VRP, introduced by Dantzig and Fulkerson in 1959, is a problem of minimizing routes' cost for a number of vehicles, whose task is to serve customer demands. Other constraints, like vehicles' capacity (CVRP), time windows (VRPTW) may be added to the basic problem to meet the demands of real-life applications [Cor+02].
Unlike the classical VRP, where all customers require the same service, a central assumption in the PDP, sometimes also denoted as VRPPD [PDH08], is that there are two different types of services that can be performed at a customer location, pick-up (origin or pick-up location) or delivery (destination or drop-off location) [SS95]. The same types of additional constraints are applied as in the VRP case.
DARP is concerned with transportation of people, and thus should take into account not only operator cost, but also quality of service from user perspective [CL07], most often in the form of maximum ride time limit for a passenger (time between the passenger's pick-up and drop-off), time window for departure or/and arrival, maximum detour time (difference between the real time and shortest path time). Other features may include a number of vehicles (single or multi-vehicle), their capacity (homogeneous or heterogeneous), maximum route time, and mode of service operation (static or dynamic).

2.1 DARP classification

Standard DARP defined by Cordeau and Laporte in [CL03b] is a problem of designing minimum-cost routes for a number of passengers in a graph with nodes corresponding to pickup and delivery locations. Each edge has an associated cost added to the total cost of the solution if the edge is a part of a route. Other features of the standard definition include depot node, which should be the start and the end node of each route, limited maximum route duration and user ride time, time window for service start and homogeneous vehicles with limited capacity.

Early surveys by Parragh et al. [PDH08] and Cordeau and Laporte [CL07]

classify DARPs into 1) single-vehicle or multi-vehicle and 2) static or dynamic. The first category is self-explanatory. In the case of the static DARP, all information about user requests is known before the start of an operation. If some information becomes available after the start of operation and the decision-maker may respond to the newly revealed information, the problem is considered dynamic.

Molenbruch et al. [MBC17] base their classification on three types of extensions to standard DARP, arising from real-life problems. The first type is service design characteristics, such as user, vehicle or driver heterogeneity or additional routing properties. The second type represents different single and multi-objective optimization functions. The last category focuses on dynamic and stohastic problems with respect to information about demand and vehicle travel times.

One of the latest surveys on DARP [Ho+18] analyzes recent developments between 2007-2017. Their classification, based on completeness of information and certainty (or perfectness of information), includes two categories 1) static or dynamic and 2) deterministic or stochastic. Other features like the number of vehicles, number of depots, time window, capacity, ride time, route duration, selectivity, fleet heterogeneity, and the number of objective functions are considered inside those four groups.

2.2 DARP solution methods

Starting from 1980 research on algorithmic aspects of on-demand transportation has been steadily growing, and a number of solution methods were developed for different types of DARP ranging from simple static instances with one depot to large-scale dynamic problems.

2.2.1 Exact methods

Exact methods for DARPs belongs to Branch-and-Bound family, and include Branch-and-Cut, Branch-and-Price, and Branch-and-Price-and-Cut algorithms. Exact methods mainly work with deterministic static single-objective DARP instances with tens of requests.

Branch-and-Cut

A Branch-and-Cut (BC) is a modification of the Branch-and-Bound method with cutting planes added to the problems in the Branch-and-Bound tree. The use of the BC algorithm for static single depot DARP was first introduced by Cordeau [Cor06]. Cuts are inequalities representing the problem's constraints. Violations of those inequalities are checked at each node of the tree. Those cuts tighten the LP-relaxations thus increasing chances of finding integer solutions. Last works on exact methods include different preprocessing techniques helping to reduce solution space [CL07; RCL07].

Branch-and-Price

Branch-and-Price (BP) method is also based on Branch-and-Bound but focuses on column generation instead of cutting planes. To apply BP to DARP, a problem should first be reformulated into a restricted master problem, in which a set of variables (or columns) is removed from LP relaxation to reduce computations, and a pricing sub-problem, where columns may be generated and added to the restricted master problem to tighten the relaxation. In the works that use the BP approach for DARPs, the master problem is mainly to optimize the objective function while the task of the subproblem is to generate routes for vehicles like in [Gar+11].

Branch-and-Cut-and-Price

Branch-and-Cut-and-Price (BPC) takes advantage of both Branch-and-Cut and Branch-and-Price by adding cutting planes to the LP relaxations during the procedure. It handles the reduced problem with columns generated by solving the subproblem and tightening the bounds for the LP relaxation. In [GI15] BPC was used to derive an effective column-generation formulation for DARPs with dynamic time windows.

2.2.2 Approximate methods

General VRP and DARP are proved to be NP-hard [LK81; jKS98], and exact methods can only solve small artificial instances (up to 100 customers) [RC09; Cor06]. In the search for scalable methods able to find sub-optimal but good solutions in a reasonable amount of time, more attention during the last decades was focused on approximate methods including various heuristics and metaheuristics.

Insertion Heuristic

One of the first heuristics for the multi-vehicle DARP was the Insertion Heuristic by Jaw et al. [Jaw+86]. It selects requests ordered by pick-up time and inserts them one by one into vehicles' routes in the way that minimizes possible increase of objective function. The model includes pickup and delivery time windows, maximum ride time, and, additionally, non-empty vehicles are not allowed to be idle. Even though it is not as efficient as metaheuristics, it is fast and simple, and different extensions [CC07; Häm11] of greedy insertion heuristic are still used, especially in dynamic DARPs [BCJ14; MZ16]. We are using an implementation of Insertion Heuristic in Chapter 4 as a benchmark for our solution.

Tabu Search

Tabu Search (TS) is a variation of local search[1], which uses tabu list (list of states temporally excluded from search space) to avoid cycling [GL97]. TS is able to produce good quality solutions for DARPs and at the same time is flexible with respect to different constraints and objective functions [CL03a].

Variable Neighborhood Search

Another variation of local search successfully applied to DARP is Variable Neighborhood Search (VNS) proposed by Mladenović and Hansen [MH97]. This metaheuristic includes a systematic change of neighborhoods in different phases by introducing randomly generated solutions [Par+09].

Adaptive Large Neighborhood search

In the Adaptive Large neighborhood search (ALNS), at each iteration of local search, a part of the solution is destroyed and then rebuilt back into a complete solution (in DARP, n requests are first removed from the solution, and then reinserted back). The algorithm may choose between different removal and insertion heuristics. The choice of heuristic depends on its past performance [RP06; BK16].

2.2.3 Hybrid algorithms

Hybrid algorithms became a growing trend during the last ten years. The first type of such method is a combination of different metaheuristics executed either sequentially or one inside another. Second type is combination of metaheuristics (VNS, ALNS) and linear programming [Par+09; MBC17; MZ16; GD16; Pim+17].
In Table 2.1 we summarize results for the maximum size of a problem solved by different methods. Approximate methods are able to solve larger problems as compared to exact methods but it is still far from real-life city-scale problems that are mainly addressed in works on large-scale dynamic transportation systems.

2.3 Large-scale dynamic systems

Dynamic ridesharing is defined as dynamically assigning vehicles with empty seats to passenger in response to a passenger's request in real time. A method designed for dynamic DARPs should produce a solution for large-scale input fast enough, otherwise, it will induce additional delay or may even completely freeze vehicles' operations. Results for maximum problem size are shown in Table 2.1.

[1] A local (or neighborhood) search starts from a candidate solution and then iteratively moves to one of neighbor solution using only information about the solutions in the neighborhood of the current one.

Method	Problem size (requests)
Insertion Heuristic [Jaw+86]	2617
Clustering, column generation, and scheduling [Ioa+95]	2445
Insertion, inter-route exchanges, secondary objective for diversification [06]	2000
Regret based insertion heuristic [DD04]	1000
Parallel insertion followed by inter- and intra-route exchanges and tabu threshold [TV97]	312
Tabu search with vertex reinsertion [CL03a]	295

Table 2.1: Approximate methods with maximum solved problem size

2.3.1 Hybrid simulated annealing

Hybrid simulated annealing (HSA) [JJP15] is a metaheuristic designed for large search spaces. SA includes periodic re-optimization for requests arrived during the previous time period. It is a stochastic relaxation method that iteratively improves an initial solution. At each iteration, the current state may be replaced by a randomly generated candidate. Random neighbor generation consists of a standard move (moves a passenger schedule from one random vehicle to another) and a swap (swaps two or more schedules between randomly selected cars) operations. A new solution is then either accepted or denied based on objective function value. The method was tested during a simulation on the synthetically generated dataset and yields higher profit without a significant increase in passenger inconvenience.

Vehicle-Group Assignment

Vehicle-group assignment (VGA) [ČA18] method finds an optimal solution by first generating all feasible groups of requests for each vehicle and then finding an optimal assignment of those groups to the vehicles. This formulation turns out to be beneficial for settings with tight time constraints, due to the fact that such constraints eliminate infeasible groups (containing requests

Method	Problem size [request]
Hybrid simulated annealing [JJP15]	18000
Vehicle group assignment, [ČA18] / Pairwise shareability graph [Alo+17]	427/ 3.000.000
Predictive routing [AWR17]	460.700

Table 2.2: Approximate methods with solved problem size

that are too far from the vehicle). Group (set of requests) is feasible for the vehicle if it can serve all requests without violating constraints. An iterative group generation procedure is based on the observation, that all subgroups of a feasible group must be feasible as well. It first generates groups of size 1 (containing single request), then of size 2, 3, and up to vehicle capacity constraint. After all possible groups for each vehicle have been generated, they are assigned to individual vehicles in such a way that every request is served by exactly one vehicle. The assignment problem is then transformed into a binary integer linear program, and solved. The method can compute a set of representative Pareto-optimal system plans to achieve different trade-offs between the cost of operation and user discomfort. VGA is a more concise reformulation of shareability graph construction proposed by Alonso-Mora et al. [Alo+17] that is simpler to implement. For the second stage, vehicle assignment, both methods rely on ILP.

Predictive routing

The solution takes into account the predicted demand. During the preprocessing step, the number of requests between different origin-destination pairs is estimated for different weekdays and times of the day. The probability distribution computed from historical data is used to predict future requests. At each iteration predicted requests are added to the demand pool, trips are greedily assigned to vehicles, and optimal assignment is computed. After that, idle vehicles are rebalanced. The method was tested on New York City taxi trip data for one week with a fleet of 1000, 2000, and 3000 vehicles and capacities of two and four passengers, and use of predicted demand and rebalancing reduced average waiting time by 1 minute, and average travel time by 1.5 minutes [AWR17].

As only methods designed for online problems offers enough scalability to solve problems consisting of tens to thousands of request, our solution described in Chapter 4 is a combination of VGA from [ČA18] and bipartite matching from [Vaz+18] and [AWR17] with some changes for the static case.

Chapter 3

Problem description

Our problem is static deterministic multi-vehicle DARP with the homogeneous fleet, pickup time window and detour time limit without rejection option. Thus the goal is to design routes and schedules for all user requests minimizing total travel time of all vehicles.

3.1 Definitions

We are considering a transportation network G, a set of transportation requests R for a certain period of time, and a set of vehicles V.
Transportation network is a directed connected graph $G = (N, E, \delta)$, where N is the set of nodes corresponding to road junctions, $E \subseteq N \times N$ is the set of edges corresponding to road links, and δ denotes cost function. The cost function $\delta : E \to \mathbb{R}_{\geq 0}$ for each directed edge $e(n_i, n_j) \in E$ is defined as shortest travel time between nodes from $n_i \in N$ to $n_j \in N$.
Transportation requests R is a set of individual requests for transportation from one spatial location to another arrived at a certain time. Request $r \in R$ is defined as tuple of pick-up and request actions (r_i^p, r_i^d) with $r_i^p = (t_i^p, n_i^p), r_i^d = (t_i^d, n_i^d)$, where t_i^p represents pick-up time, $n_i^p \in N$ pick-up location, t_i^d drop-off time, and $n_i^d \in N$ drop-off location, respectively. Pick-up time means the earliest time t_i^p at which the passenger can be picked up at location n_i^p (the time when the request was created by the passenger). The drop-off time is the earliest possible time of dropping off the passenger defined as $t_i^d = t_i^p + \delta(e(n_i^p, n_i^d))$.
Each $v_i \in V$ has maximum capacity $Q \in \mathbb{R}_{\geq 0}$ (number of passenger seats), and the fleet is assumed to be homogeneous with respect to capacity. The number of vehicles is not limited.
Our task is to find a feasible set of transportation plans P for vehicles where $p \in P$ is a sequence of pick-up and drop-off events

$$p = ((n_1^p, t_1^p), (n_i^p, t_i^p), \ldots, (n_1^d, t_1^d), \ldots, (n_j^p, t_j^p), (n_i^d, t_i^d), (n_j^d, t_j^d))$$

serviced by exactly one vehicle $v \in V$ that minimizes total cost across all vehicle plans for V. Feasible set of plans contains each request $(r_i^p, r_i^d) \in R$ exactly once. For any r_i, the pick-up should happen before the drop-off, and both (r_i^p, r_i^d) should belong to the same plan.

Classical mathematical models, with three-index variables by Cordeau [Cor06], and two-index by Ropke [RCL07], are Integer Linear Program (ILP) formulations directly used to find the optimal solution. We do not use ILP formulation for the whole problem due to the number of requests, only for the subproblems that will be discussed later in Section 4.1.2.
As compared with the standard model, we do not have constraints limiting maximal ride time and route duration. To control passenger discomfort, we, instead, introduce constant $\alpha \in \mathbb{R}_{\geq 0}$, maximum prolongation for the request, i.e., the difference between real arrival time and earliest possible arrival time. The constant α is used in time-window constraints 3.5 and 3.6. The standard model addresses a single-depot case where each plan must start and end at the same depot node[1]. In our formulation, we use virtual depot, a special node n_0 with the following property $\delta(n_0, n) = \lambda \; \forall n \in N$, so that the vehicle starting from that node can reach any other node in a fixed constant amount of time $\lambda \in \mathbb{R}_{\geq 0}$. Our formulation is as follows:

minimize

$$\sum_{p \in P} \delta(p) \tag{3.1}$$

subject to

$$\exists! p_v \in P \qquad \forall v \in V \tag{3.2}$$

$$\exists! v \in V : r \in p_v \qquad \forall r \in R \tag{3.3}$$

$$\widehat{t_r^p} < \widehat{t_r^d} \qquad \forall r \in R \tag{3.4}$$

$$t_r^p \leq \widehat{t_r^p} \leq t_r^p + \alpha \qquad \forall r \in R \; i > 1 \tag{3.5}$$

$$t_r^d \leq \widehat{t_r^d} \leq t_r^d + \alpha \qquad \forall r \in R \tag{3.6}$$

$$t_1^p \leq \widehat{t_1^p} \leq t_1^p + \lambda \tag{3.7}$$

where constraints 3.2 enforces that each vehicle v has exactly one plan p, constraint 3.3 enforces that each request r is served exactly once. Constraint 3.4 ensures correct pick-up and drop-off order, i.e., the actual pick-up of each trip $\widehat{t_r^p}$ must precede the actual drop-off time $\widehat{t_r^d}$. Constraints 3.5 and 3.6 are pick-up and drop-off time windows, they define the earliest and latest possible arrival time for pick-ups and drop-off, respectively. The last constraint 3.7 enforces constant travel time to the first pick-up node in the vehicle's plan.

[1] Depot is just one of the graph nodes as any other node.

Chapter 4

Solution

Our solution is an integral part of AmodSim[1], set of Java packages for traffic simulation build on top of multi-agent simulation framework AgentPolis[2]. The solution is divided into two stages. First, we find optimal plans for requests inside some longer time interval (batch), and, second, when all request batches are processed, connect optimal plans into final vehicle schedules.

4.1 Intra-batch optimization

Each batch contains sorted requests that arrived during some period. To find shared trips inside the batch we use the Vehicle-Group Assignment method (VGA) presented in [ČA18]. This method finds an optimal solution for a set of trips by generating groups (4.1.1) for vehicles and subsequently choosing a cost-minimal subset from those groups (4.1.2).

4.1.1 Group Generation

Group Generation procedure generates set of groups $G \subseteq R$ for each vehicle $v \in V$ consisting of all cost-minimal groups of requests that are feasible for the vehicle. Group $g \in G$ is feasible for the vehicle if that vehicle can serve all requests from the group subject to usual precedence, capacity, and time-window constraints (3.4, 3.5, 3.6).

Cost of the plan P equals to total time vehicle needs to travel to serve all requests $r \in P$, and $\sigma(\hat{P})$ denotes the minimal cost for the group (cost of the optimal plan $\hat{P}$ made of requests from the group). In the original setting, the whole method works with requests arrived during the last 30 seconds (batch) of simulation and active requests that arrived earlier. Requests are considered active while the current simulation time is less than the request's latest possible pick-up time. After that, they are discarded. Discarded requests incur the additional cost (penalization).
In our case, when each trip has a guaranteed vehicle with constant travel time from the depot, the group generation procedure will return identical sets

[1] `https://gitlab.fel.cvut.cz/fiedlda1/amod-to-agentpolis`
[2] `https://github.com/aicenter/agentpolis`

of groups for all vehicles, thus we actually need to generate groups only once for each batch of requests. Another modification is the feasibility check for generated groups. In the online case, the car cannot arrive at the request's pick-up node before that request has arrived because of the simulation time. There is no need to check the lower bound of the pick-up time window (earliest possible arrival). In the offline case, there is no notion of the current time and we need to check both bounds of the time window so that a vehicle arrives at the given request pick-up location no earlier than the request time.
Moreover, to simplify computations online implementation assumes that all requests from the same batch have identical request time (earliest possible arrival) equal to the current simulation time at the start of batch processing. Requests in the batch are ordered and for each trip, its cost becomes bigger than real travel time by the time passed from the start of the batch till the actual request time. If we compare two groups consisting of the same requests that would benefit the one where the first request arrived earlier. With a longer batch interval, the start time for each trip inside the batch should be set individually to avoid such overestimation that would lead to a result that is far from optimal.

4.1.2 Vehicle-Group Assignment

A vehicle-group assignment in [ČA18] is a mapping $a : V \to P$ from the set of vehicles to the set of feasible plans generated at the previous step. The minimum-cost vehicle-group assignment is then obtained by solving an Integer Linear Program (ILP).
In the online case with the limited number of vehicles and simulation time, each vehicle has some location in space at the start of a new batch. It also has different travel times from that location pick-up locations of different requests. In our setting, we have constant travel time to the first pick-up node and the unlimited number of vehicles[3]. Concerning vehicle-group assignment that means we need to check that each request is chosen exactly one, i.e. to enforce constraint 3.3 saying that each request belongs to exactly one plan selected for the optimal assignment. Our problem is translated into the following ILP:

$$\begin{aligned}
&\operatorname*{argmin}_{\{x_G\}} \sum_{g \in G} x_g \cdot \sigma(g), \\
&\text{subject to :} \\
&\qquad \sum_{g \in G} x_g = 1 \\
&\sum_{g \in G} x_g \cdot \mathbb{1}_g(r) = 1 \qquad \forall r \in R \\
&\qquad x_g \in \{0, 1\} \qquad \forall g \in G,
\end{aligned}$$

where $\mathbb{1}_g(r)$ is the indicator function, i.e. $\mathbb{1}_g(r) = 1$ if $r \in g$ and $\mathbb{1}_g(r) = 0$ otherwise, and constraints enforce that each group and each request are

[3] We can describe it as if we had a dedicated depot with one car two minutes away from each pick-up node in the batch

chosen exactly once.
After group generation is done, generated groups are passed to ILP solver which returns a set of optimal group plans $\widehat{P}$ for the given batch of requests. At this stage, we do not limit the number of vehicles setting it equal to the number of requests in the batch.

4.2 Plan chaining

Finally, optimal plans from consecutive batches are connected into a complete plan with an assigned vehicle. For that, we use a slightly modified approach from [Vaz+18], which addresses the minimum fleet-size problem for on-demand mobility without ridesharing. The introduced approach finds minimum number of vehicles needed to serve all requests under the assumptions that 1) a vehicle is available at each pick-up location l_i^p on or before time t_i^p and 2) connection time, i.e., travel time between drop-off n_i^d and next pick-up n_{i+1}^p, is at most σ. It also uses the notion of a vehicle shareability network. This is a directed graph, where nodes correspond to trips and the existence of an edge between two nodes indicates that the two incident requests can be consecutively served by a single vehicle. This vehicle shareability network is then translated into an exact formulation of the minimum fleet problem as a minimum path cover problem on directed acyclic graphs. The solution is a sequence of trips to be served for each vehicle in the minimum fleet.
Our modifications are as follows: 1) we use of plans which may include more than one request instead of individual requests and 2) instead of assuming that each pick-up r_i^p has an available vehicle at location n_i^p at time t_i^p or before, we assume that vehicle is available at time $t_i^p + \alpha$ or before (3.5).
To construct vehicle shareability network we consider the whole plan P as single request with pick-up time equal to the pick-up time of plan's first request and drop-off time equal to the drop-off time of plan's last request, i.e., for the plan with n requests $t^p = t^p_{r_1}, t^d = t^d_{r_n},\ r_1, r_n \in R,\ |P| = n$. The shareability network is defined as a bipartite graph $B = (N^d \cup N^p, E)$, where node n_i^p corresponds to plan's p_i pick-up node, and node n_j^d to plan's p_j drop-off node. Directed edge $e(n_i^d, n_j^p) \in E$ exists if and only if $t_i^d + \delta(n_i^d, n_j^p) \leq t_j^p + \alpha$ (3.5) and $t_j^p - t_i^d \leq \sigma$. If such a link exists, two incident plans can be served by the same vehicle. Diagram in Fig. 4.2 illustrates the network construction. Graph B is acyclic due to the acyclic nature of time (for proof, see [Vaz+18]), and the minimum path cover for B may be found using the Hopcroft-Karp algorithm (Algorithm 1) for bipartite matching that takes bipartite graph and produces a maximum cardinality matching - a set of as many edges as possible with the property that no two edges share an end node.

A vertex that is not the endpoint of an edge in some partial matching M is called a free vertex. The basic concept that the algorithm relies on is that of an augmenting path, a path that starts at a free vertex, ends at a free vertex, and alternates between unmatched and matched edges within the path.

The resulting optimal matching is an ordered sequence of plans where the plan's position in the given sequence encodes drop-off action (action

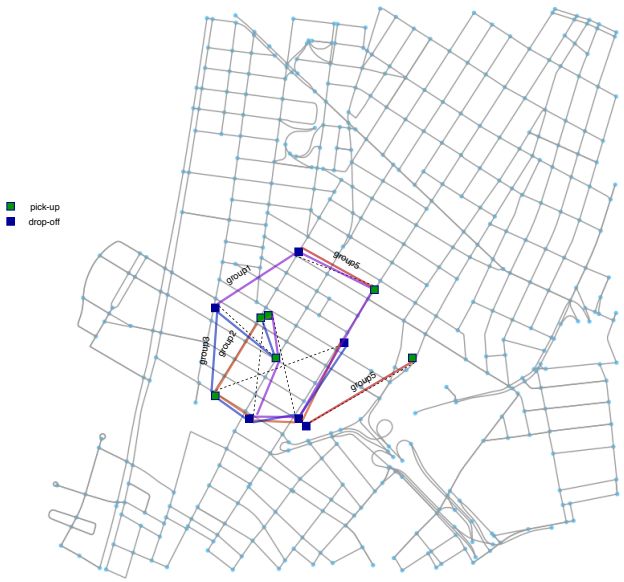

Figure 4.1: Process of group generation

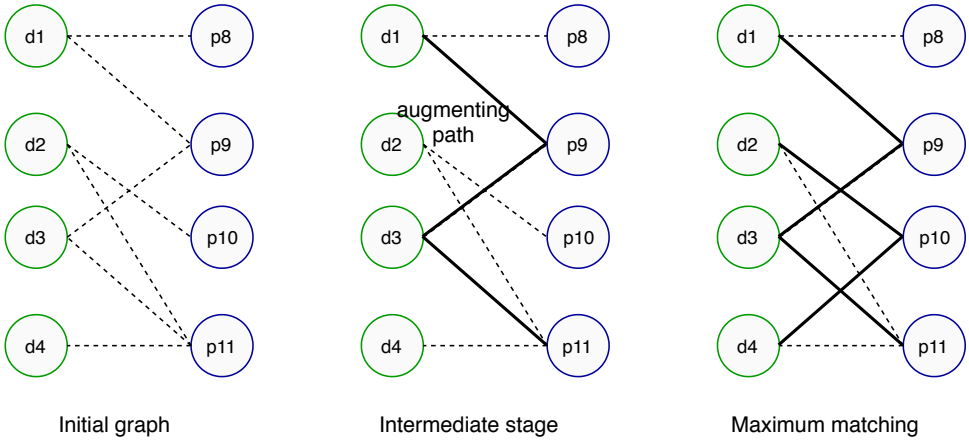

Figure 4.2: Maximum matching in bipartite graph

Algorithm 1 Hopcroft-Karp

$G \leftarrow (N^d \cup N^p, E)$
$M \leftarrow \emptyset$
while $P \neq \emptyset$ **do**
    $P \leftarrow \{P_1, P_2, \ldots, P_k\}$ ▷ maximal set of vertex-disjoint shortest augmenting paths
    $M \leftarrow M \oplus (P_1 \cup P_2 \cup \cdots \cup P_k)$
return $M \subseteq E$ ▷ optimal matching

request's id), and value in that position is next pick-up action's id. Those paths connecting individual plans combined with plans' request data gives the final solution $P = (p_{v_1}, p_{v_2}, \ldots, p_{v_m})$. Each $p_v \in P$ needs a dedicated vehicle, $|V| = |P| = m$. Below is a fragment of one real plan generated during evaluation:

$$
\begin{aligned}
p_{v_1} = (\\
&(n_1^p, t_1^p),\ (n_2^p, t_2^p),\ (n_1^d, t_1^d),\ (n_2^d, t_2^d), \ldots\\
&, (n_3^p, t_3^p),\ (n_3^d, t_3^d), \ldots\\
&, (n_4^p, t_4^p),\ (n_5^p, t_5^p),\ (n_6^p, t_6^p),\ (n_7^p, t_7^p),\\
&(n_6^d, t_6^d),\ (n_7^d, t_7^d),\ (n_4^d, t_4^d),\ (n_5^d, t_5^d),\\
&, \ldots)
\end{aligned}
$$

This segment consists of three group plans of size 2, size 1, and size 4.

Chapter 5

Evaluation

In this chapter, we will describe data used for testing and evaluation of our approach presented in Chapter 4, analyze how different model parameters influence the quality of the final solution, and compare results with and without ridesharing produced by the proposed method and that of Insertion Heuristic described in Section 2.2.2.

5.1 Data

5.1.1 Road network

The graph used for evaluation is shown in Fig. 5.1. It was build from OpenStreetMap data for bounding box with latitude from 40.7255 °W to 40.70768 °W and longitude from -73.99411 °N to -74.01756 °N.
The graph contains 5633 nodes and 12353 edges and is strongly connected. Nodes represent road junctions in the central region of New York City (mainly in Manhattan). The graph (or most of its part) is a regular grid with rectangular cells with an average side length of 268 meters and average travel time of only 19 seconds (Fig. 5.2).

5.1.2 Request data

New York City Taxi and Limousine Commission[1] provides public access to trip record data from New York taxi service providers on their homepage [New19]. Up to 2016, TLC datasets contained exact pick-up and drop-off coordinates. For evaluation, we took one day from the year 2015. During the year 2015 so-called yellow taxis served from 100.000 to 500.000 trips daily (See Fig. 5.3), mostly in central regions of the city, including Manhattan, so even a relatively small area of central Manhattan accounts to several thousand requests during one day of operation.

During the prepossessing stage, raw request data is filtered and transformed following the definition from 3.1. Requests located outside of our area of interest together with requests for which both pick-up and drop-off locations were mapped to the same graph node (3.4) are discarded. Valid requests are

[1] `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

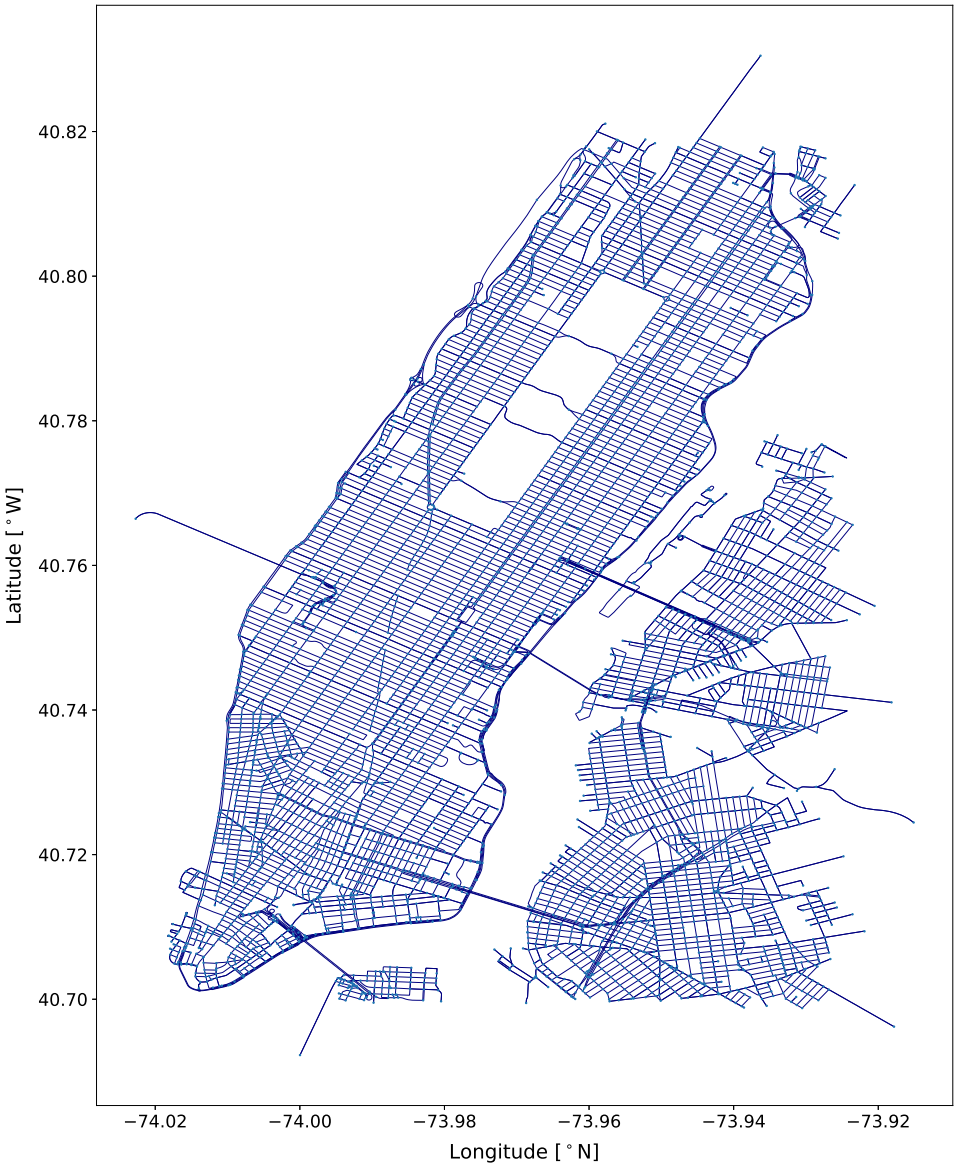

Figure 5.1: Road network

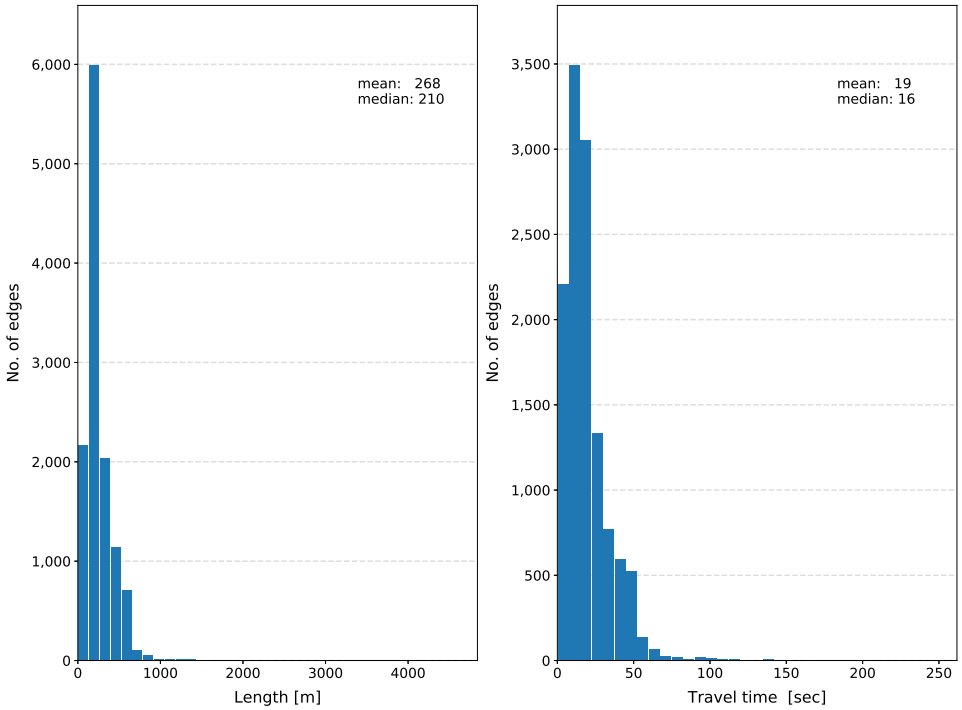

Figure 5.2: Road link lengths and travel times from OpenStreetMap data

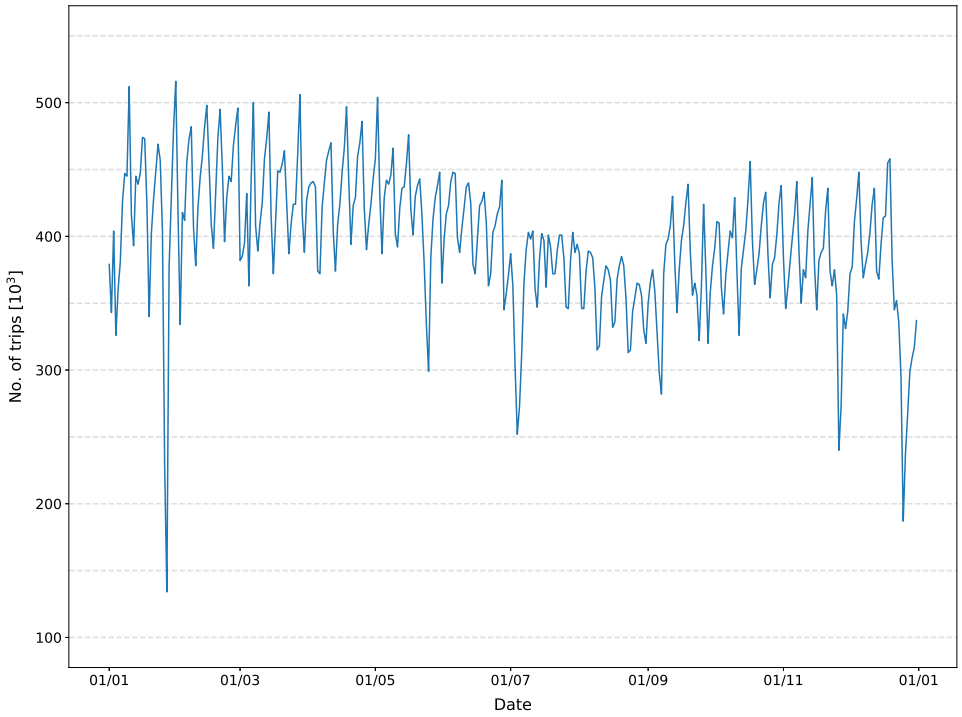

Figure 5.3: Trip counts for yellow taxis through 2015

ordered by start time (this order used to uniquely identify request through the rest of computations) and divided into batches.
Dataset1 consists of 23.485 requests from the central part of New York (Manhattan and part of Brooklyn) area between 08:00 and 10:00 (morning rush hours). Trips are relatively short: if in reality taxis always drove with maximal allowed speed for each road link without any delays, an average taxi trip would last less than 5 minutes (Fig. 5.4). As can be seen from Fig. 5.5, most of the demand inside the given coordinates originates from Midtown Manhattan.

Dataset2 is prepared similarly and contains 200.064 requests from 00:00 to 23:59 of the same day as dataset1. The mean estimated trip length calculated over the whole day is about 7 minutes. Demand density varies throughout the day: it significantly decreases during night time and makes two peaks during morning and evening rush hours.

5.2 Experiments

We first studied how different parameters (batch interval and σ) influence final solution on dataset1, and then evaluated our solution on dataset2. In addition to minimized objective function which in our case is total distance travelled by all vehicles, we also measure total delay (or passenger discomfort) defined as

$$\sum_{p_v \in P} \sum_{r \in p_v} \widehat{t_r^d} - t_r^p - \delta(n_r^p, n_r^d).$$

As a baseline solution for comparison we have chosen Insertion Heuristic 2.2.2 which is still widely used as a part of more complex solutions ([Fie+18; JJP15]).

5.2.1 Dataset1: 2 hours

As compared to [Vaz+18] we have chosen 10 minute interconnection time, i.e., travel time from plan's last drop-off node to next plan's first pick-up node, for Chaining[2]. Our graph is smaller, and trips are shorter as if compared to whole New York City [3]. Starting from 10 minutes fleet size stops decreasing (See Table 5.1 and Fig. 5.6), and further increase of interconnection time does not influence final solution, because requests with pick-up nodes that lie too far away from the previous drop-off are not chosen as immediate continuation in Hopcroft-Karp 1 procedure. To avoid meaningless computations, particularly in the case of dataset2, for the rest of the experiments sigma is set to 600 seconds.

Another parameter estimated on dataset1 is the length of the batch interval. It would be natural to expect that longer optimization periods would give bet-

[2] σ is not involved in group generation and vehicle group assignment

[3] One reason, is that their origin and destination nodes are limited inside a smaller area, another possible explanation is that Manhattan has higher taxi fares per unit of distance compared to other districts of New York

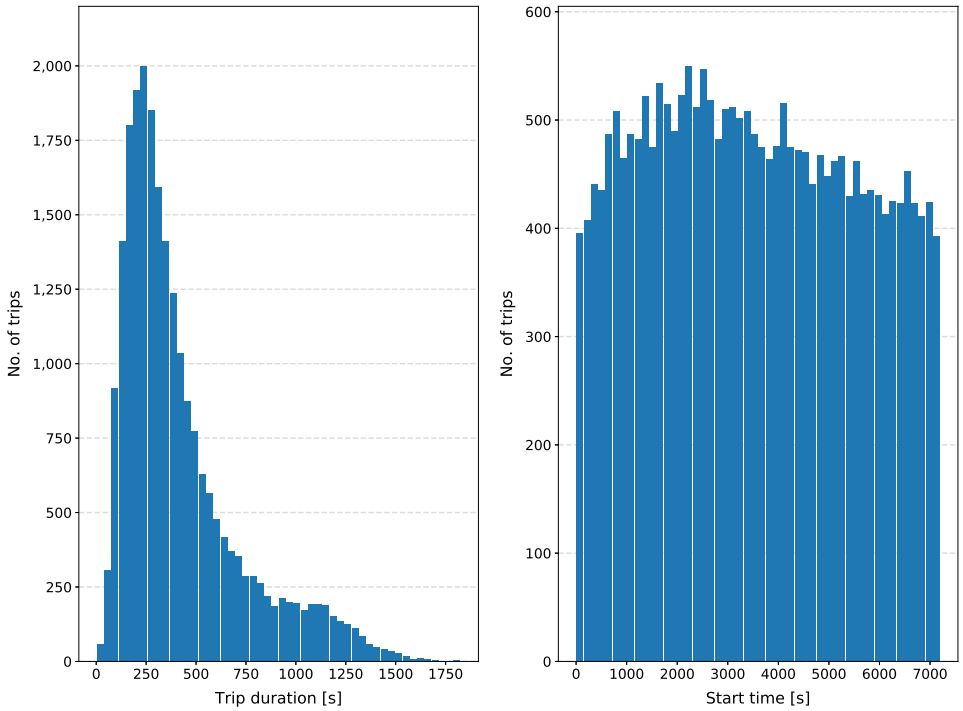

Figure 5.4: Requests duration along the shortest route and requests start times

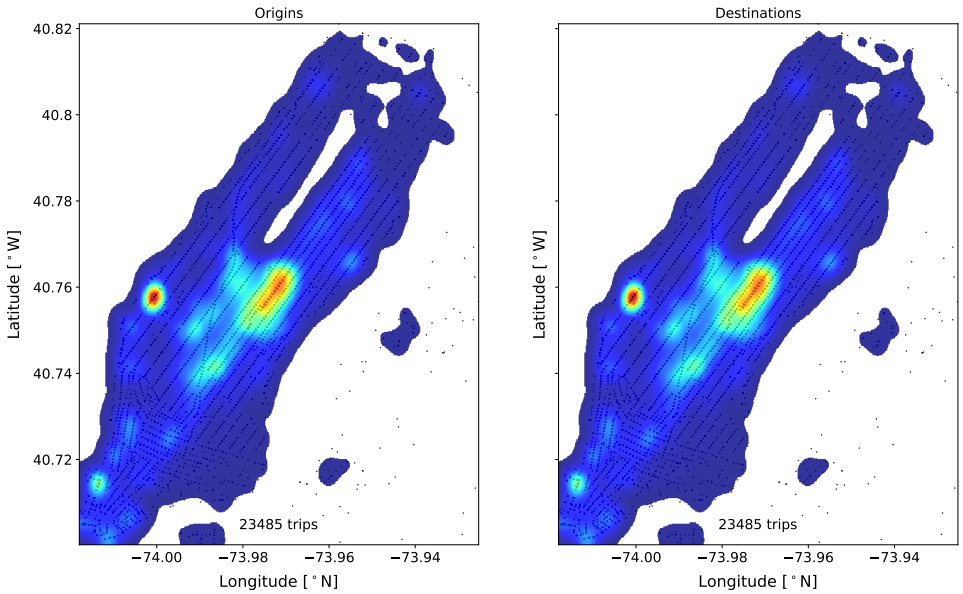

Figure 5.5: Pick-up (Origins) and drop-off (Destinations) nodes' location and density

Sigma, [sec]	**Fleet size**
120	2013
180	2012
240	1990
300	1565
600	1546
900	1546
∞	1546

Table 5.1: Fleet size for different values of σ

Batch interval, [sec]	**Total time travelled,** [hour]	**Time with passengers,** [hour]	**Time with passengers,** % of total
30	3463.48	2638.24	76
60	3309.39	2536.28	77
120	3042.30	2368.18	78
180	2878.56	2248.75	78
240	2736.00	2151.64	79
300	2637.52	2082.81	79

Table 5.2: Dataset1. Results for different batch intervals

ter solutions. On the other hand, the process of group generation 4.1.1 in our setting appeared to be very memory expensive. The implementation we used is primarily designed for online mode and works in 30-second batches [ČA18]. As batch duration grows together with the number of requests in the batch, a number of groups generated during by the procedure may start to grow exponentially[4]. Another reason is the character of demand where individual requests are located relatively close to each other in time and space (See Fig. 5.7 and Fig. 5.8). Even with an adaptation that handles all necessary constraints the longest interval we were able to compute is 5 minutes (that is around 1000 requests). Results for batch intervals from 30 seconds up to 5 minutes are shown in 5.2 and 5.9.

Finally, we measured total time travelled, time travelled with passengers, and total delay for different group sizes from 1 (no ride-sharing) to 4 and compared our results to that of Insertion Heuristic. Final parameters for evaluation are listed in Table 5.3. Travel times are shown in Table 5.4, with more detailed data about fleet size and groups in Fig. 5.10.

The use of ridesharing reduced total time travelled by all vehicles by 1045 hours (28%), with a proportional decrease in fleet size (by 624 vehicles) during 2 hours of operation with about 23.000 requests, and incurred average delay of 0.71 minutes per request. The total additional delay accumulated by all

[4]Moreover, it overestimates plan's cost by time elapsed from start of the batch to plan's first pick-up action and does not need to enforce some of our time constraints.

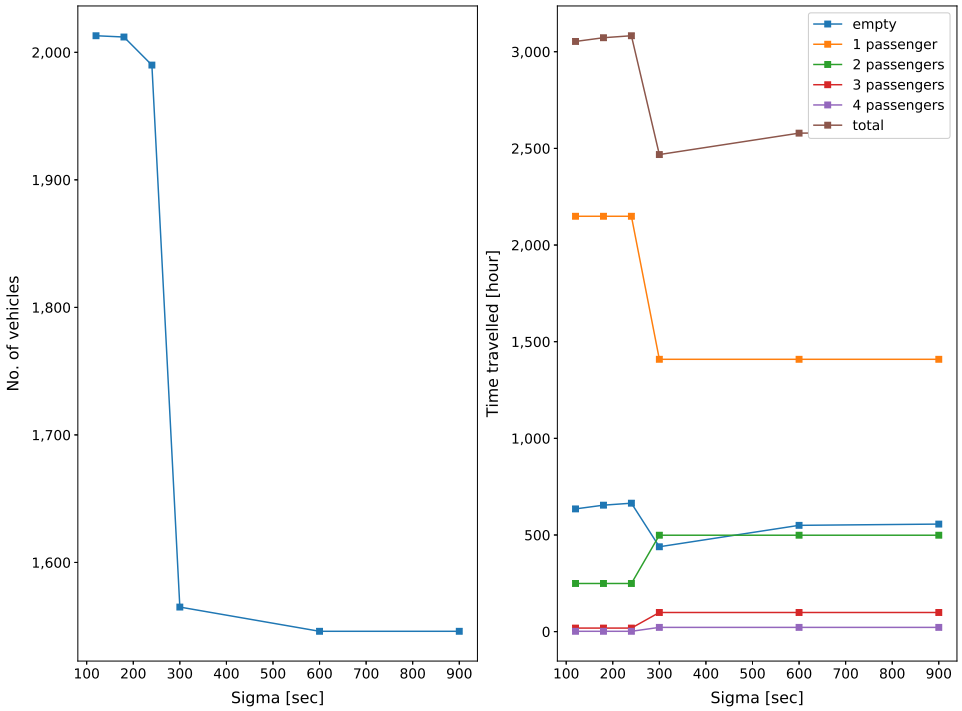

Figure 5.6: Dataset1. Fleet size and time travelled for different values of σ

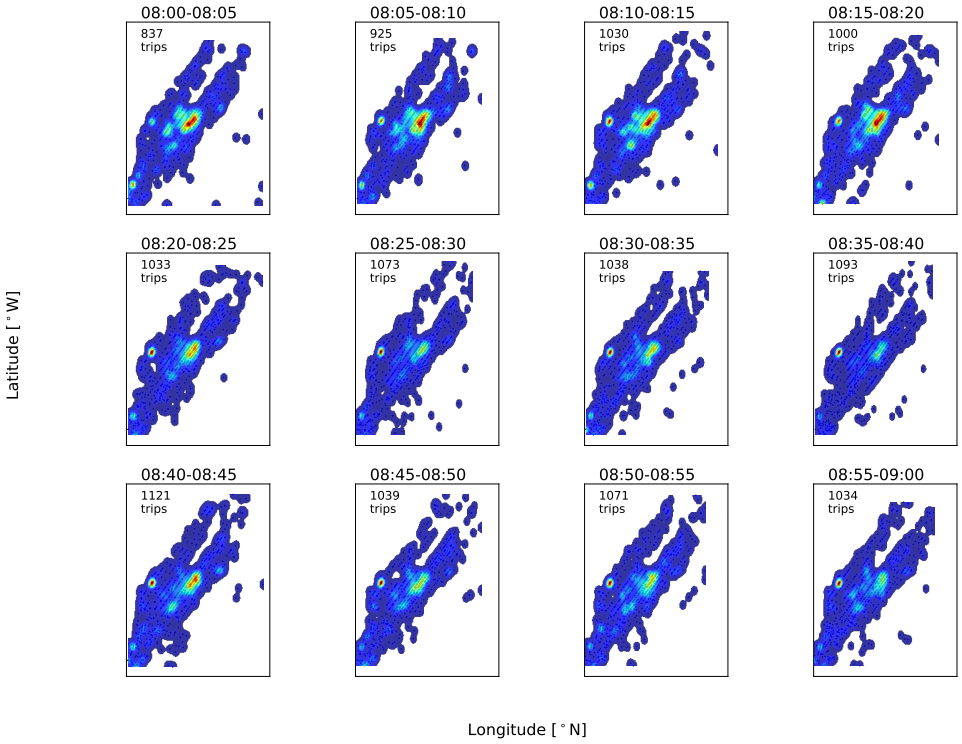

Figure 5.7: Pick-up nodes' locations and density in 5 minute intervals

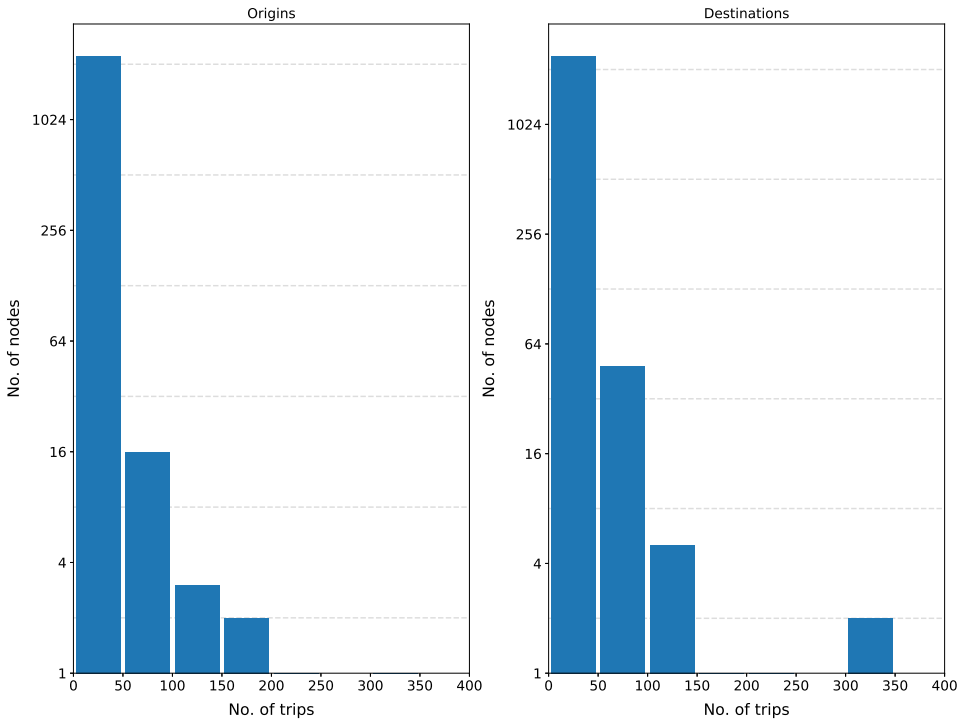

Figure 5.8: Node histograms by number of pick-ups (Origins) and drop-offs (Destinations)

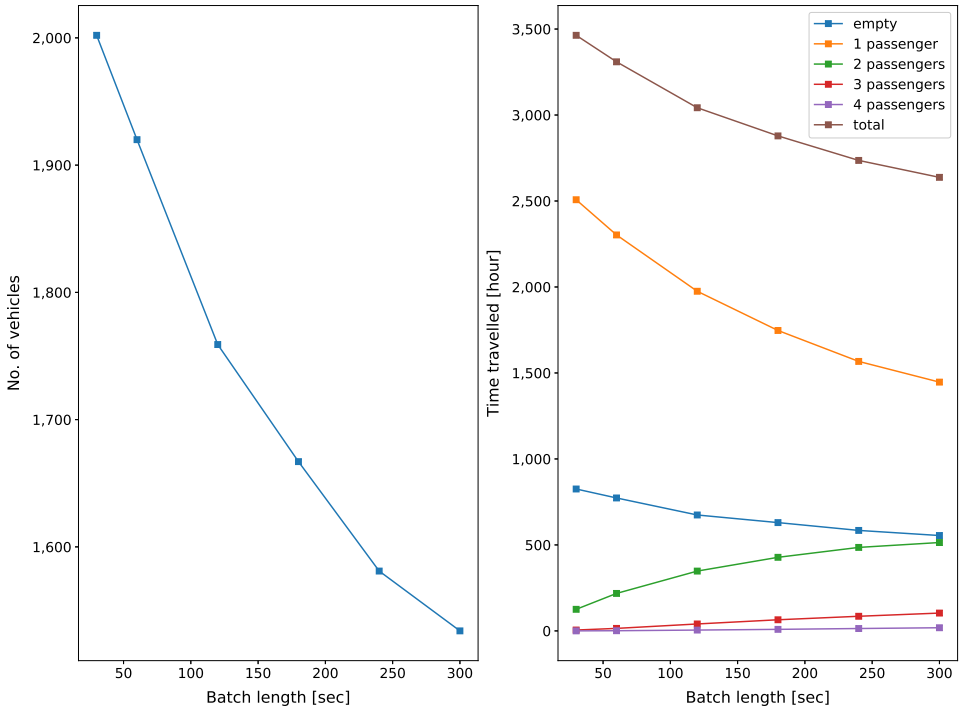

Figure 5.9: Dataset1. Fleet size and time travelled for different batch intervals

Parameter	Value	Meaning
σ	600 sec	maximum connection time between plans
λ	60 sec	time to 1st pick-up in vehicle plan
α	120 sec	max delay (prolongation)
Q	1 - 4	vehicle capacity
	300 sec	batch interval

Table 5.3: Experiment parameters

Vehicle capacity	Total time travelled, [hour]	Time with passengers, [hour]	Time with passengers, % of total
1	3682.57	2761.20	75
2	2763.03	2159.52	78
3	2656.68	2090.48	79
4	2637.52	2082.81	79

Table 5.4: Dataset1. Results for different vehicle capacities

vehicles is 214 hours[5] (Fig. 5.11).

Insertion Heuristic on dataset1 showed better results in terms of total time, 2194 hours as compared to 2530 by our solution but for the cost of increased passenger discomfort. For IH resulting value of the objective function, total travel time, is 13% lower and the total incurred delay is 76% higher. Delay per request for IH is equal to 83 s which means that an average trip became more than 30% longer. Results are reported in Table 5.5 and Fig. 5.12.

5.2.2 Dataset2: 24 hours

Results of experiment are summarized in Table 5.6 and Fig. 5.13. For 24 hour interval and about 200.000 request with the same settings (5.3), total time travelled decreased by 25% with average delay of 0.56 minute per request (7.6% of average trip length) and additional total delay of 1768,8 hours by all vehicles during whole period of operation (Fig. 5.11). On dataset2 Insertion Heuristic yields 2% better results for total travelled time (25790 hours compared to 26357 of our solution). On the other hand, an average delay for Insertion Heuristic remains the same (1.2 minutes), while an average delay of the solution decreased to 0.56 minutes per request, resulting in more

[5]Chaining without ride-sharing also accumulates delay because new vehicle always needs some fixed time to reach first pick-up node.

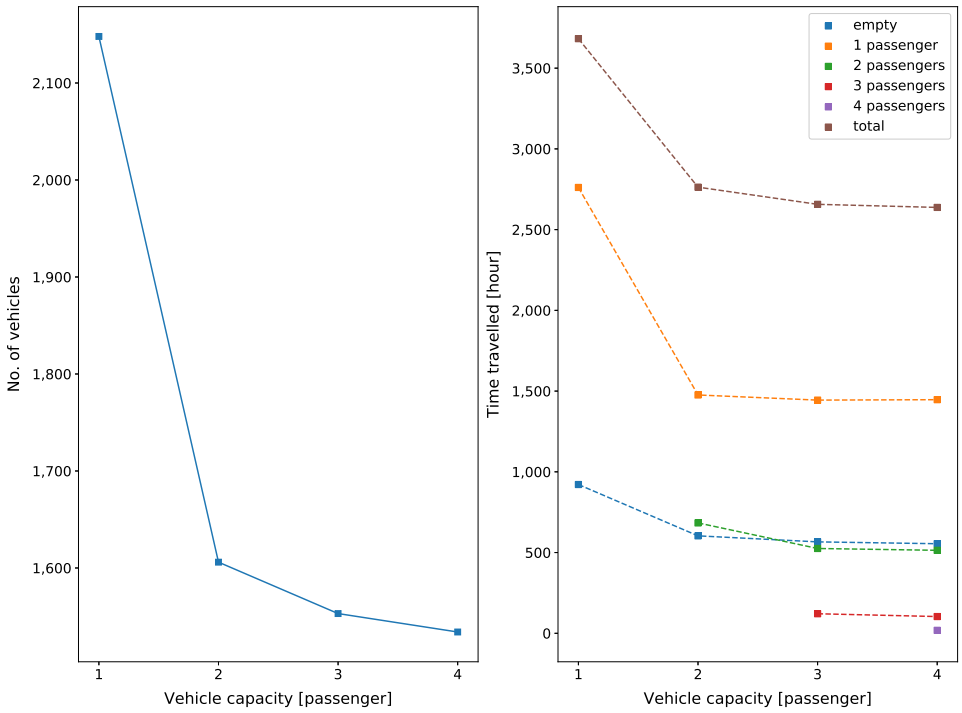

Figure 5.10: Dataset1. Fleet size and time travelled with different capacities

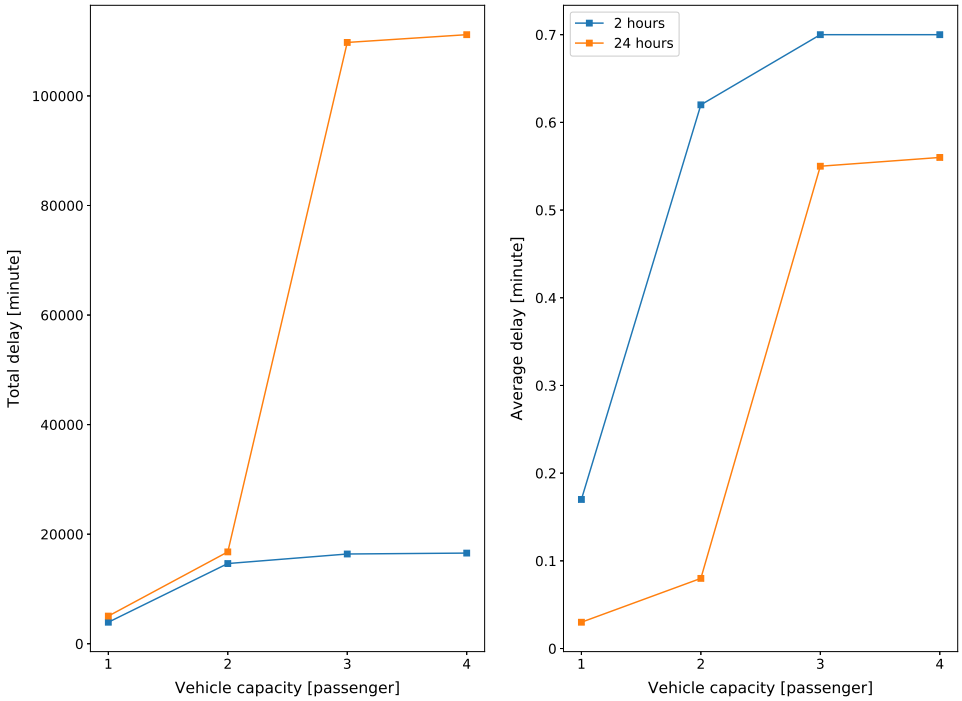

Figure 5.11: Average delay per request and total delay for vehicle capacity from 1 to 4 for dataset1 (2 hours) and dataset2 (24 hours)

Method	Total time travelled, [hour]	Time with passengers, [hour]	Total delay, [hour]	Average delay, [min]
Insertion Heuristic	2193.97	1850.58	483	1.23
Chaining	3682.57	2761.20	61.38	0.17
VGA+ Chaining	2637.52	2082.81	275.81	0.70

Table 5.5: Dataset1. Travelled times and delays for Insertion Heuristic, Chaining ($\sigma = 10$ min), and VGA+Chaining ($\sigma = 10$ min, 5-minute batch interval)

Vehicle capacity	Total time travelled, [hour]	Time with passengers, [hour]	Time with passengers, % of total
1	3682.57	2761.20	75
2	2763.03	2159.52	78
3	2656.68	2090.48	79
4	2637.52	2082.81	79

Table 5.6: Dataset2. Total travelled time and time travelled with passengers for vehicle capacity from 1 to 4

than more two times smaller total delay. Results are reported in Table 5.7 and Fig. 5.14.

5.3 Discussion

VGA optimally solves minimum-cost vehicle-group assignment problem for the processed batch of requests. Non-optimal solutions in the online case may appear due to the overestimated cost for some requests, as was described in Section 4.1.1. In our case, this overestimation was eliminated but we do not process all requests at once. On one single batch, VGA shows better results, e.g., for batch with 500 its average result in terms of travel time is 1.8% better and for 900 requests - 7.8% as compared with Insertion Heuristic. The number of used vehicles also was smaller in solutions produced by VGA. As seen from results, groups generated by our solution have a maximum length equal to maximum vehicle capacity and always have the same form: from 1 to 4 pick-ups following by the same number of drop-offs, e.g., $p_1, p_2, p_3, d_2, d_1, d_3, p_4, d_4, \ldots$. Groups generated by IH tend to be longer and contain mixed groups, e.g., $p_1, p_2, d_2, p_3, d_1, p_4, d_3, d_4, \ldots$.
To get better solutions, we need to increase the batch interval. And the further increase is not possible due to technical reasons. The current implementation of VGA available from Amodsim is designed for dynamic online problems.

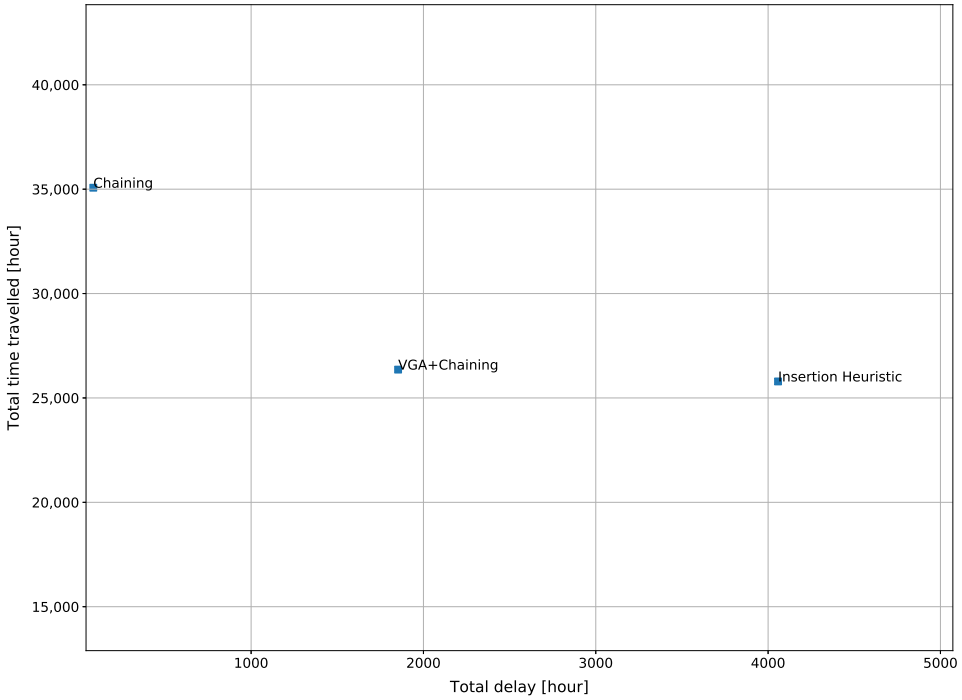

Figure 5.12: Dataset1. Total time and total delay for Insertion Heuristic, Chaining (σ = 10 min), and VGA+Chaining (σ = 10 min, 5-minute batch interval)

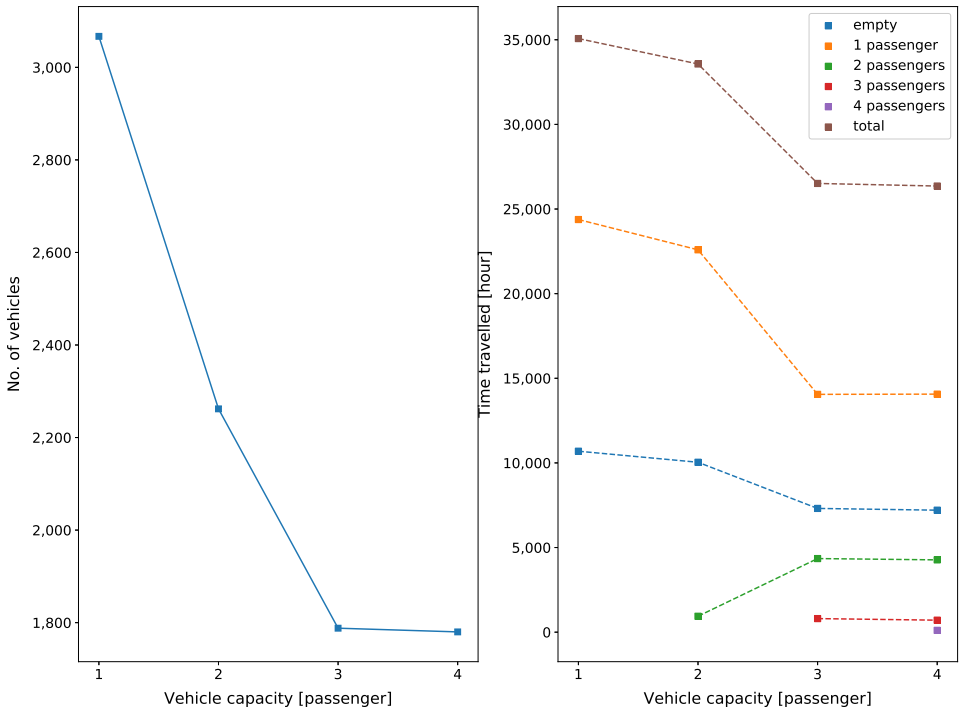

Figure 5.13: Dataset2. Travelled times for vehicle capacity from 1 to 4

Method	Total time travelled, [hour]	Time with passengers, [hour]	Total delay, [hour]	Average delay, [min]
Insertion Heuristic	25790.23	16824.52	4057.70	1.22
Chaining	35070.03	24379.00	84.16	0.03
VGA+ Chaining	26357.32	19150.38	1853.00	0.56

Table 5.7: Dataset2. Travelled times and delays for Insertion Heuristic, Chaining ($\sigma = 10$ min), and VGA+Chaining ($\sigma = 10$ min, 5 minute batch interval)

In the offline case, its bottleneck is the group generation procedure that appears to be extremely memory consumptive. The longest batch interval we were able to compute is 7 minutes: this additional 2-minute increase in optimization interval reduced travel time by 6.3%.
Our evaluation also showed that on a bigger dataset relative difference in travel time between the solution and IH becomes smaller. IH, described in Section 2.2.2, is a greedy construction heuristic that does not re-optimizes previously computed plans. With longer plans, IH moves further away from optimality, and its scalability over the long time horizon is doubtful.

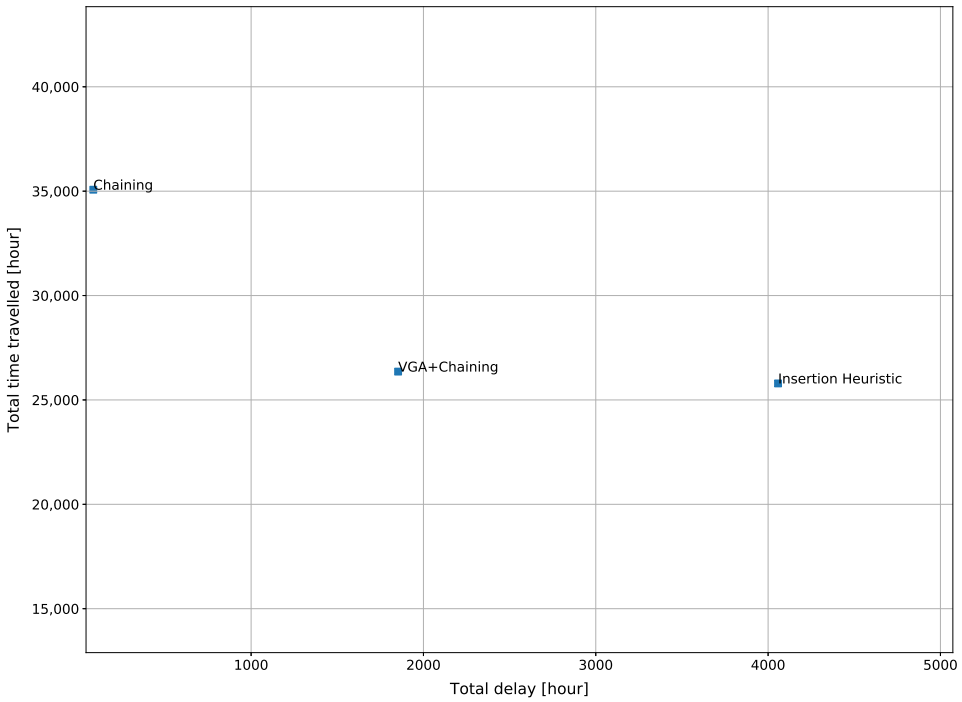

Figure 5.14: Dataset2. Total time and total delay for Insertion Heuristic, Chaining (σ = 10 min), and VGA+Chaining (σ = 10 min, 5 minute batch interval)

Chapter 6

Conclusion

In this work, we presented a method for request assignment, inspired by [ČA18], and vehicle routing from [Vaz+18]. Both methods are originally designed to handle large-scale dynamic online problems. The proposed solution consists of two steps. The first step (VGA) is generating all possible minimum-cost group plans for a limited time interval and choosing from all generated groups a subset that minimizes total travel time under the constraint that each request is chosen exactly once. During the second stage (Chaining) optimal plans from consecutive batches are connected into final plans with assigned vehicles.

We tested our solution on the demand data consisting of 23.000 requests and evaluated it on a larger dataset of approximately 200.000 requests. We experimentally showed that the adoption of ridesharing may reduce total time travelled by all vehicles along with the fleet size by about 25% at the cost of a small passenger inconvenience, i.e., increase in average trip length. As we already mentioned in Section 6, on a single batch, without Chaining, VGA alone always produces solutions with a better value of the objective function in comparison with Insertion Heuristic used in many algorithms for large-scale dynamic scheduling problems. During our evaluation combination of VGA and Chaining showed slightly worse results in terms of total travel time. The reason for that is the batch-processing of requests. When batch duration becomes longer, travel time and fleet size decrease, an additional increase of batch length by 2 minutes produced 6.3% better solution. But current implementation of the group generation procedure described in 4.1.1 designed for use in online simulations is very memory-consumptive and further experiments would require completely new implementation specifically designed to work in that setting. On the other hand, Insertion Heuristic has significantly worse results from the user point of view and has limited scalability for offline usage. As the number of requests in vehicles' plans grows, IH moves further from optimality because it does not re-optimize previously computed plans.

It could be interesting to apply VGA for known DARP benchmark instances, but it also requires significant changes in the current implementation (or the new one) because of the totally different input data representations.

Appendix A

Bibliography

[06] "A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints". In: *European Journal of Operational Research* 174.2 (2006), pp. 1117–1139.

[AWR17] J. Alonso-Mora, A. Wallar, and D. Rus. "Predictive routing for autonomous mobility-on-demand systems with ride-sharing". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Sept. 2017), pp. 3583–3590.

[Alo+17] J. Alonso-Mora et al. "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment". In: *Proceedings of the National Academy of Sciences* 114.3 (Jan. 2017), pp. 462–467.

[BCJ14] K. Braekers, A. Caris, and G. K. Janssens. "Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots". In: *Transportation Research Part B: Methodological* 67 (2014), pp. 166–186.

[BK16] K. Braekers and A. Kovacs. "A multi-period dial-a-ride problem with driver consistency". In: *Transportation Research Part B: Methodological* 94 (Dec. 2016), pp. 355–377.

[CC07] R. Calvo and A. Colorni. "An effective and fast heuristic for the Dial-a-Ride problem". In: *4OR* 5 (Apr. 2007), pp. 61–73.

[ČA18] M. Čáp and J. Alonso-Mora. "Multi-Objective Analysis of Ridesharing in Automated Mobility-on-Demand". In: *Robotics: Science and Systems* XIV (2018).

[Cor06] J.-F. Cordeau. "A Branch-and-Cut Algorithm for the Dial-a-Ride Problem". In: *Operations Research* 54.3 (May 2006), pp. 573–586.

[CL03a] J.-F. Cordeau and G. Laporte. "A tabu search heuristic for the static multi-vehicle dial-a-ride problem". In: *Transportation Research B* 37 (2003), pp. 579–594.

[CL03b] J.-F. Cordeau and G. Laporte. "The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms". In: *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1.2 (June 2003), pp. 89–101.

[CL07] J.-F. Cordeau and G. Laporte. “The dial-a-ride problem: Models and algorithms”. In: *Annals of Operations Research* 153.1 (Sept. 2007), pp. 29–46.

[Cor+02] J.-F. Cordeau et al. “A Guide to Vehicle Routing Heuristics”. In: *Journal of the Operational Research Society* 53 (May 2002), pp. 512–522.

[DD04] Marco Diana and Maged Dessouky. “Dessouky, M.M.: A New Regret Insertion Heuristic for Solving Large-Scale Dial-a-Ride Problems with Time Windows. Transportation Research, Part B 38, 539-557”. In: 38 (July 2004), pp. 539–557.

[Fie+18] D. Fiedler et al. “The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague”. In: *CoRR* abs/1807.03352 (2018), pp. 1173–1178.

[Fur+13] M. Furuhata et al. “Ridesharing: The state-of-the-art and future directions”. In: *Transportation Research Part B: Methodological* 57 (2013), pp. 28–46.

[Gar+11] T. Garaix et al. “Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation”. In: *Computers Operations Research* 38 (Oct. 2011), pp. 1435–1442.

[GL97] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[GD16] T. Gschwind and M. Drexl. *Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem*. Working Papers 1624. Gutenberg School of Management and Economics, Johannes Gutenberg-Universität Mainz, Dec. 2016.

[GI15] T. Gschwind and S. Irnich. “Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem”. In: *Transportation Science* 49 (2015), pp. 335–354.

[Häm11] L. Häme. “An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows”. In: *European Journal of Operational Research* 209.1 (2011), pp. 11–22.

[Ho+18] S. C. Ho et al. “A survey of dial-a-ride problems: Literature review and recent developments”. In: *Transportation Research Part B: Methodological* 111.C (2018), pp. 395–421.

[INR18] INRIX. *INRIX 2018 Global Traffic Scorecard*. 2018. URL: `http://inrix.com/scorecard/` (visited on 10/30/2019).

[Ioa+95] I. Ioachim et al. “A Request Clustering Algorithm in Door-to-Door Transportation”. In: *Transportation Science* 29 (Feb. 1995), pp. 63–78.

[Jaw+86] J.-J. Jaw et al. “A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows”. In: *Transportation Research Part B: Methodological* 20.3 (1986), pp. 243–257.

[jKS98] J. W. Baugh jr., G. K. R. Kakivaya, and J. R. Stone. "Intractability of the Dial-a-Ride problem and a multiobjective solution using simulated annealing". In: *Engineering Optimization* 30.2 (1998), pp. 91–123.

[JJP15] J. Jung, R. Jayakrishnan, and J. Y. Park. "Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing". In: *Computer-Aided Civil and Infrastructure Engineering* 31 (June 2015).

[LK81] J. K. Lenstra and A. H. G. R. Kan. "Complexity of vehicle routing and scheduling problems". In: *Networks* 11.2 (1981), pp. 221–227.

[MZ16] M. Mahmoudi and X. Zhou. "Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state-space-time network representations". In: *Transportation Research Part B: Methodological* 89 (2016), pp. 19–42.

[MH97] N. Mladenović and P. Hansen. "Variable neighborhood search". In: *Computers Operations Research* 24.11 (1997), pp. 1097–1100.

[MBC17] Y. Molenbruch, K. Braekers, and A. Caris. "Typology and literature review for dial-a-ride problems". In: *Annals of Operations Research* 259.1 (Dec. 2017), pp. 295–325.

[New19] City of New York. *TLC Trip Record Data*. 2019. URL: `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page` (visited on 10/30/2019).

[PDH08] S. N. Parragh, K. F. Doerner, and R. F. Hartl. "A survey on pickup and delivery problems". In: *Journal fur Betriebswirtschaft* 58.1 (Apr. 2008), pp. 21–51.

[Par+09] S. N. Parragh et al. "A heuristic two-phase solution approach for the multi-objective dial-a-ride problem". In: *Networks* 54.4 (2009), pp. 227–242.

[Pim+17] V. Pimenta et al. "Models and algorithms for reliability-oriented Dial-a-Ride with autonomous electric vehicles". In: *European Journal of Operational Research* 257.2 (2017), pp. 601–613.

[RC09] S. Ropke and J.-F. Cordeau. "Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* 43.3 (June 2009), pp. 267–286.

[RCL07] S. Ropke, J.-F. Cordeau, and G. Laporte. "Models and branch-and-cut algorithms for pickup and delivery problems with time windows". In: *Networks* 49.4 (2007), pp. 258–272.

[RP06] S. Ropke and D. Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* 40 (Nov. 2006), pp. 455–472.

[SS95] M. W. P. Savelsbergh and M. Sol. “The General Pickup and Delivery Problem”. In: *Transportation Science* 29.1 (1995), pp. 17–29.

[Tom18] TomTom. *New York in the Traffic Index*. 2018. URL: https://www.tomtom.com/en_gb/traffic-index/new-york-traffic (visited on 10/30/2019).

[TV97] Paolo Toth and Daniele Vigo. “Heuristic Algorithms for the Handicapped Persons Transportation Problem”. In: *Transportation Science* 31.1 (1997), pp. 60–71.

[Vaz+18] M. Vazifeh et al. “Addressing the minimum fleet problem in on-demand urban mobility”. In: *Nature* 557 (May 2018). DOI: 10.1038/s41586-018-0095-1.

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kholkovskaia Olga** Personal ID number: **464308**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Scalable Offline Ridesharing Algorithm

Bachelor's thesis title in Czech:

Škálovatelný algoritmus pro offline ridesharing

Guidelines:

1. Explore the state-of-the-art methods for ridesharing with focus on offline ridesharing.
2. Create a short review of ridesharing algorithms. Categorize the methods (optimal/heuristic, online/offline, ...) and compare them (scalability, efficiency, ...).
3. Design a method for offline ridesharing that can solve metropolitan-scale scenarios.
4. Implement the designed method and prepare it for the integration to other software.
5. Compare the method with at least one other heuristic ridesharing method. Analyze the differences in scalability and ridesharing efficiency.

Bibliography / sources:

[1] D. Fiedler, M. Čertický, J. Alonso-Mora and M. Čáp, "The Impact of Ridesharing in Mobility-on-Demand Systems: Simulation Case Study in Prague," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 1173-1178.
[2] Jung, Jaeyoung & Jayakrishnan, R & Young Park, Ji. (2015). Dynamic Shared-Taxi Dispatch Algorithm with Hybrid Simulated Annealing. Computer-Aided Civil and Infrastructure Engineering. 31. 10.1111/mice.12157.
[3] M. Cap and J. Alonso-Mora, "Multi-Objective Analysis of Ridesharing in Automated Mobility-on-Demand," in Robotics: Science and Systems XIV, 2018.
[4] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," in 2017 IEEE RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 3583–3590.
[5] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing Vehicle Distributions and Fleet Sizes for Mobility-on-Demand," presented at the IEEE International Conference on Robotics and Automation (ICRA), 2019, p. 7.

Name and workplace of bachelor's thesis supervisor:

Ing. David Fiedler, Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **12.06.2019** Deadline for bachelor thesis submission: **07.01.2020**

Assignment valid until: **19.02.2021**

Ing. David Fiedler Supervisor's signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature