Dynamic Car-Passenger Matching of Online and Reservation Requests

Marvin Erdmann
BMW AG
Mobility Services
Parkring 19
85748 Garching, Germany
Email: marvin.erdmann@bmw.de

Florian Dandl
Bundeswehr University Munich
Institute for Intelligent Transportation Systems
Werner-Heisenberg-Weg 39
85779 Neubiberg, Germany
Email: florian.dandl@unibw.de
ORCiD: 0000-0003-3706-9725

Bernd Kaltenhäuser
Baden-Wuerttemberg Cooperative State University
Department of Technical Management
Friedrich-Ebert-Str. 30
78054 Villingen-Schwenningen, Germany
Email: bernd.kaltenhaeuser@dhbw-vs.de
ORCiD: 0000-0001-9786-6465

Klaus Bogenberger
Bundeswehr University Munich
Institute for Intelligent Transportation Systems
Werner-Heisenberg-Weg 39
85779 Neubiberg, Germany
Email: klaus.bogenberger@unibw.de

Word count: 7111 words text + 1 tables x 250 words (each) = 7361 words
Number figures: 5

Submission Date: August 1, 2019

ABSTRACT

Dynamic Car-Passenger Matching is a variant of the Dial-a-Ride Problem (DaRP) and a crucial part of operating an On-Demand Mobility (ODM) service. The ODM fleet management algorithm has to be able to find close-to-optimal solutions quick and reliable in order to provide a high quality service to the users of the service. In this work, an approach is presented that combines customer requests for immediate pick-ups (online) and reservations for pick-ups in the near future. This approach is based on the method of Global Optimization with Time Windows (GOTW), which is able to benefit from both a very fast accept/reject response with an estimated pick-up time window generated by a greedy Nearest Neighbor Policy (NNP) heuristic as well as the optimization potential of a periodically executed Tabu Search. This work also introduces List-Based Assignments (LBAs) as a replacement of the initial NNP in order to shorten computational times of optimization without loosing any feasible solution. The solutions provided by the algorithm using this new approach generate up to 67.7% less empty mileage compared to the solutions found by NNP. This advantage outweighs the slightly higher number of rejections and longer waiting times for most of the evaluated scenarios. Especially in settings considering reservations, the new methodology clearly outperforms the benchmark algorithm by up to around 36% in terms of the overall quality of solution.

Keywords: Dynamic Car-Passenger Matching, On-Demand Mobility, Global Optimization with Time Windows, List-Based Assignments, Reservations

INTRODUCTION

In today's urban traffic travelers face many problems and inconveniences, among others congestion and the lack of space due to many parking cars. On-Demand Mobility (ODM) is an increasingly popular concept that tries to tackle these challenges by allowing people to profit from individual urban mobility without owning a car. The enhanced use of shared mobility services and the increased utilization of vehicles in cities would lead to less space occupied by parking cars without loosing the benefits of always available mobility.

To operate a fleet with many cars and customers, a management algorithm is necessary to match service requests and vehicles of the fleet dynamically. It must be able to quickly find a reliable and time efficient solution for the whole system. Such a problem is called Dial-a-Ride-Problem (DaRP)(1). As a generalization of the Traveling Salesman Problem it is NP-hard, which means finding optimal solutions requires exponentially rising computational time with increasing problem size. Due to the dynanism of the problem and the necessity of short response time to customers of the service, heuristics are often used to circumvent these long computational times. An often used method is called Nearest Neighbor Policy (NNP), which matches cars to new customers immediately based on the respective arrival times. Instead of local optimization (NNP), more sophisticated approaches like metaheuristics aim to optimize the whole fleet, but avoid exponential computational times by smart exploration of the solution space. Osman and Kelly (2) defined a metaheuristic as '[...] an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions'.

The significance of metaheuristics to solve the DaRP for the ODM use case led to an increasing interest in this research area recently. The static version of the problem has been discussed in (3) by Cordeau and Laporte in which they use a metaheuristic procedure called Tabu Seach (TS). Brandão used TS for the heterogeneous fixed fleet vehicle routing problem in (4). In (5), Prodhon and Prins present and compare the most important heuristics and metaheuristics for the Vehicle Routing Problem, in which they emphasize the good performances of TS algorithms. Pandi et al. (6) used a GPU-accelerated TS algorithm to solve the DaRP, which produced comparable results upto 30 times faster than a single-core CPU-based TS algorithm.

The models used in simulations to test the algorithms vary significantly, since there are countless specifics an ODM service may offer to its customers. Alonso-Mora et al. (7) considered the ride-sharing use case for immediate-pick-up requests using constraint optimization. Hyland and Mahmassani (8) focus on the ride-hailing use case. They compare six assignment strategies, four of which assign requests only once and could therefore provide information rather quickly to the customers. In (9), Sheridan et al. propose a dynamic nearest neighbor policy to find good solutions for highly dynamic problems in a very fast way. Though, none of the mentioned papers considers both reservations and online requests as possible ways for customers to request a pick-up.

However, the incorporation of reservations into the dynamic optimization of online requests is an important task for mobility services offering both options to their customers, like Uber (10) and Lyft (11). Short accept/reject response times are as important for reservations as for online requests. Therefore, this initial decision should be made by a fast heuristic and the optimization should only take place periodically instead of each time a new request occurs. Hence, this work is based on an approach called Global Optimization with Time Windows (12) which combines the quick response times of NNP with the optimization potential of TS. The new approach considers reservations as well as online request to find nearly optimal solutions for the combined problem. It focuses on the so called ride-hailing use case in which at most one customer occupies a car.

The objective of this work is to present a method to combine ODM requests for immediate pick-ups (online) with reservation requests in a close-to-optimal way using TS. To guarantee a fast response to the customers, a heuristic based on the idea of a NNP is used to determine if a request is accepted or not. The benefits of applying the presented methods in a model which uses the New York Taxi Data Set (13) will be evaluated and compared to results found using only NNP, as it is an often used standard for the problem in real world applications.

PROBLEM FORMULATION

The DaRP has been formulated in different variants since it first was established. In this work, a very common formulation is used with constraints found in e.g. (7), (8) and (14).

A solution for the DaRP is considered to be optimal if both the fleet costs and the dissatisfaction of the customers are minimal over a given period of time. As fleet costs are mainly produced by the movement and corresponding fuel consumption of vehicles in the fleet, the overall distance between all visited locations in the solution should be as short as possible. More specifically, the empty mileages between drop-off locations and pick-up locations should be minimized, as the demanded routes from a customer's pick-up location to his or her destination are independent of the assignments and should therefore not be part of the optimization.

In the dynamic version of the DaRP, new online customer requests occur eventually. These requests are not known ahead of time, so the operator can only solve the DaRP with a subset of active requests in order to find solutions shortly after customers apply for the service. Hence, this optimization of new requests should take place periodically after a certain time window t_{period} in which new requests are accepted or rejected. On the one hand, t_{period} - defined by the fleet operator - has to be short enough to guarantee quick response times to the customers. On the other hand, the longer t_{period} is chosen to last, the more requests are optimized at once, increasing the optimization potential. So, the choice of this parameter is critical.

Each time an optimization takes place, all eligible customers N with pick-up locations $i \in [1, ..., N]$ and drop-off locations $j \in [N+1, ..., 2N]$ and an ODM fleet with M vehicles at their next idle locations $m \in [2N+1, ..., 2N+M]$ are considered. The set of pick-up and drop-off locations of the customers is referred to as P and D respectively. The next idle location of a vehicle is either its actual position (if it is idle) or the drop-off location of the last request it is matched to. The set of all next idle locations is defined as C. $V = P \cup D \cup C$ is the set of vertices of an undirected graph $G = (V, A)$, with A the set of arcs containing all feasible pairs of locations (a, b), with $a, b \in V$.

If in a solution a vehicle moves from a to b, the decision variable $x^m_{a,b} = 1$, otherwise $x^m_{a,b} = 0$. Every arc (a, b) is weighted with a cost $c_{a,b}$ proportional to the distance between a and b.

In this work, the objective function consists of two terms, both of which ought to be minimized.

$$\min\left(f_{\text{obj}}\right) = \min_{x^m_{a,b}} \left(\sum_{m \in M} \sum_{a \in D \cup C} \sum_{b \in P} c_{a,b} \times x^m_{a,b} + \sum_{i \in N} d_i \left(x^m_{a,b}\right) \right), \tag{1}$$

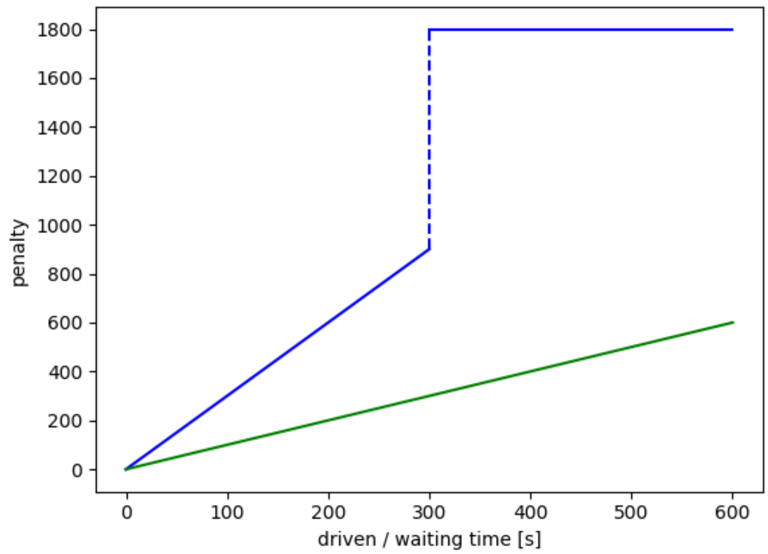


FIGURE 1 Penalty model: customer dissatisfaction $t_\mathrm{w} \times s$ if $t_\mathrm{w} \leq t_\mathrm{mw}$ or $t_\mathrm{mw} \times s \times r$ otherwise (blue); cars causing penalty equal to empty driven time in seconds (green).

The first term represents the empty driven mileage of all cars in the fleet. Each empty car is defined to produce a penalty of $\frac{1}{s}$ when on its way to pick up a customer.

The second term is the sum of all customers' dissatisfaction. The dissatisfaction d_i of each customer $i \in N$ depends on the waiting time t_w of this customer. With increasing waiting time, d_i rises linearly with slope s. In this work $s = \frac{3}{\mathrm{sec}}$, indicating the priority of customer satisfaction compared to fleet costs. If no car would be able to pick up a customer before his or her maximum waiting time t_mw, the algorithm rejects the request immediately. If a request is rejected, d_i is set to a constant value, which is r times higher than the dissatisfaction at t_mw. r is set to the value of 2, reflecting the expectation of customers not using the service in the future anymore when being rejected once. The penalty model is outlined in figure 1.

While optimizing the solution, the following constraints must be satisfied at every time:

$$\sum_{i \in C \cup D} \sum_{j \in P} x_{i,j}^m - x_{j,j+N}^m = 0 \quad \forall m \in M \tag{2}$$

$$\sum_{m \in M} x_{i,j}^m \leq 1 \quad \forall j \in P \tag{3}$$

$$t_{i,p} - t_{i,d} < 0 \quad \forall i \in N \tag{4}$$

Equation 2 ensures that if a customer $j \in N$ is picked up, the drop-off follows immediately and is done by the same car (pairing constraint). So, there is at most one customer in a car at each time (capacity constraint). In Equation 3, the constraint is formulated that every customer is served at most once. The precedence constraint is stated in equation 4 by only allowing pick-up times $t_{i,p}$ earlier than drop-off times $t_{i,d}$ for each customer $i \in N$ respectively.

Additionally, reservations are considered to be another type of service customers might request. They can be ordered some time before the desired pick-up time, which allows customers to plan trips in advance. Reservations always have to be served in time in the proposed model, which means cars need to arrive at the pick-up location no later than the requested pick-up time. If the operator cannot guarantee the service, the reservation request is rejected.

METHODOLOGY

After a short introduction to the Nearest Neighbor Policy, the Tabu Search procedure and Global Optimization with Time Windows, the methods of List-Based Assignments and the incorporation of reservation requests in online request optimization are explained in this section.

Nearest Neighbor Policy

The Nearest Neighbor Policy (NNP) is a simple heuristic designed to find feasible solutions of the DaRP very fast. Every time a new online request occurs, the distances from each of the vehicles' next idle locations to the pick-up location of the new customer are calculated and the earliest possible pick-up times are derived. If the maximum waiting time of the customer is not surpassed, the vehicle that is able to pick up the new customer the earliest is matched permanently with that request, otherwise the request is rejected. The request is immediately added to the route of the car and it moves directly to the pick-up location if it is idle.

The benefit of using the NNP is its ability to produce feasible solutions without any considerable delay as the computational time necessary to find solutions with it scales linearly with the fleet size. As it only considers the very next request, it tends to fail finding optimal solutions for the whole system. Nonetheless, in real world applications those solutions are used to operate ODM fleets. Hence, this heuristic is used as a generator of initial solutions and as the benchmark for the proposed algorithm.

Tabu Search

The concept of Tabu Search (TS) is a variant of the so called Local Search technique, which in general improves initial solutions by applying modifications (or moves) to it iteratively. The basic idea of the TS metaheuristic is to avoid local minima of the objective function value by allowing moves in a local search that produce worse solutions while forbidding already visited (tabu) solutions.

In this work, a preliminary solution is generated with the NNP following the process explained in the previous section. At the end of each optimization period after t_{period}, this becomes the initial solution of the optimization. Three types of local moves are then applied to it. The swap move switches requests (R_1,...,R_N) within the route of one car ($[R_1, R_2] \rightarrow [R_2, R_1]$), the shift move changes the position of a request from one car's route to the end of another car's route ($[R_1, R_2], [R_3, R_4] \rightarrow [R_1], [R_3, R_4, R_2]$). The third move is called interchange and exchanges the positions of two requests in two different cars' routes ($[R_1, R_2], [R_3, R_4] \rightarrow [R_1, R_4], [R_3, R_2]$). These moves are executed for all cars and pairs of cars of the fleet and the resulting solution is tested to be feasible. All feasible solutions are then defined as the neighborhood of the initial solution.

The solutions in the neighborhood of the initial solution are sorted by their value of the objective function. Beginning with the best solution, the so called Tabu List is checked, which indicates if a solution was already found recently. If so, the next solution is checked until a maximum number of solutions was checked and the search terminates. If a new solution was found before the search terminates, this solution becomes the initial solution of the next iteration. If this solution is better than the Best Solution Found (BSF) in this optimization step, it also becomes the BSF. The next iteration of optimization starts by applying local moves on the new solution. This procedure is repeated until a) the search terminates because it cannot find a solution that was not recently found or b) the search does not produce a BSF for a certain amount of iterations I_{noBSF} or c) a maximum number of iterations I_{max} had been performed or d) the time limit set by the optimization period's length is reached. After the TS terminated, the last BSF is the resulting solution of the optimization.

Global Optimization with Time Windows

Global Optimization with Time Windows (GOTW) is a method introduced in (12) which is able to benefit from both the response speed of the NNP as well as the optimization potential of the TS.

At the beginning of each optimization period, an empty list L_{opt} is set up, which will contain all requests subject to optimization at the end of the optimization period. After the simulation of all cars' movements according to their currently matched requests, at every simulation step new requests that are added to the problem are preliminary matched to a car or rejected, following the NNP. Every accepted request is added to L_{opt} and if the preliminary matched car is idle, it moves to the pick-up location starting with the next simulation step. A customer is not allowed to be rejected after his or her request was initially accepted and is not allowed to be re-scheduled in a way he or she would have to wait longer than the maximum waiting time, defined by the fleet operator. By using the NNP as an initial decision maker for accepting or rejecting a request, it is possible to send a response to the customer very quick, either a rejection or a latest pick-up time.

After adding new requests to the problem, it is checked if preliminary matched requests' pick-up locations are already reached by the respective cars. If so, those requests are matched permanently to the car, added to the actual solution and deleted from L_{opt}.

This simulation circle is executed until the end of the current optimization period. At this point, all requests in L_{opt} are subject to a global optimization using TS as described in the previous section. The found solution is considered as permanent and the customer could now be informed about the exact pick-up modalities.

Another optimization period begins by setting up a new L_{opt}. This procedure is performed periodically until the end of simulation.

List-Based Assignments

The idea of List-Based Assignments (LBAs) is to improve the initial solution by enhancing the simple NNP heuristic while decreasing the size of the solution space needed to be searched by the TS metaheuristic, without eliminating feasible solutions. A short example is presented to illustrate the approach.

As described in section *Nearest Neighbor Policy*, the NNP searches for the vehicle that is able to pick up a customer $i \in N$ fastest, once a new request is submitted. The other cars' earliest pick-up times for that particular request are not stored. When another request $j \in N$ is submitted, this car is considered to be occupied until the drop-off time of customer i, to guarantee a pick-up in time. If no other car is able to pick up customer j, this request is rejected.

With the method of LBAs though, all cars that are able to pick up a customer $i \in N$ before he or she had to wait longer than the maximum waiting time t_{mw} are added to a list $L_{\text{LBA},i}$. This list is sorted by the earliest possible pick-up time and the fastest car is matched preliminary with the request. Thus, initially the same preliminary match is made for customer i as it would have been done following the NNP.

However, when the aforementioned request $j \in N$ occurs, the preliminary matched car is not considered to be occupied until it would drop off customer i. If it is able to pick up customer j before the respective maximum waiting time and if there is another car in $L_{\text{LBA},i}$ which had not been preliminary matched to another request in the meantime and is still able to pick up customer i in time, the request of customer j can be accepted by preliminary matching it to the car, that was first sent to request i.

By doing so, more online requests can be accepted in a respective optimization period compared to only using NNP without loosing the guarantee for accepted customers to be picked up in a time shorter than the maximum waiting time and without considerable additional computational effort.

Furthermore, as only vehicles in the list $L_{\text{LBA},i}$ are able to pick up customer i, the TS can neglect solutions, in which i is matched to a vehicle that is not in $L_{\text{LBA},i}$. This decreases the number of possible solutions tremendously, which translates to a much faster optimization. As this version of the DaRP is very dynamic, this advantage is crucial for the success of the proposed algorithm in real world applications.

Optimization of Reservations

As customer $i \in N$ requesting a ride at an exact point in time in the future $t_{\text{res},i}$ wants to be informed about his or her acceptance or rejection as quick as possible, the initial decision is made by the NNP as for online requests. In contrast to online requests, matched requests are not added to the actual route of the respective car $m \in M$, but instead stored in another list, denoted as $L_{\text{res},m}$.

As explained in section *Problem Formulation*, requests in $L_{\text{res},m}$ are not part of optimization and cars

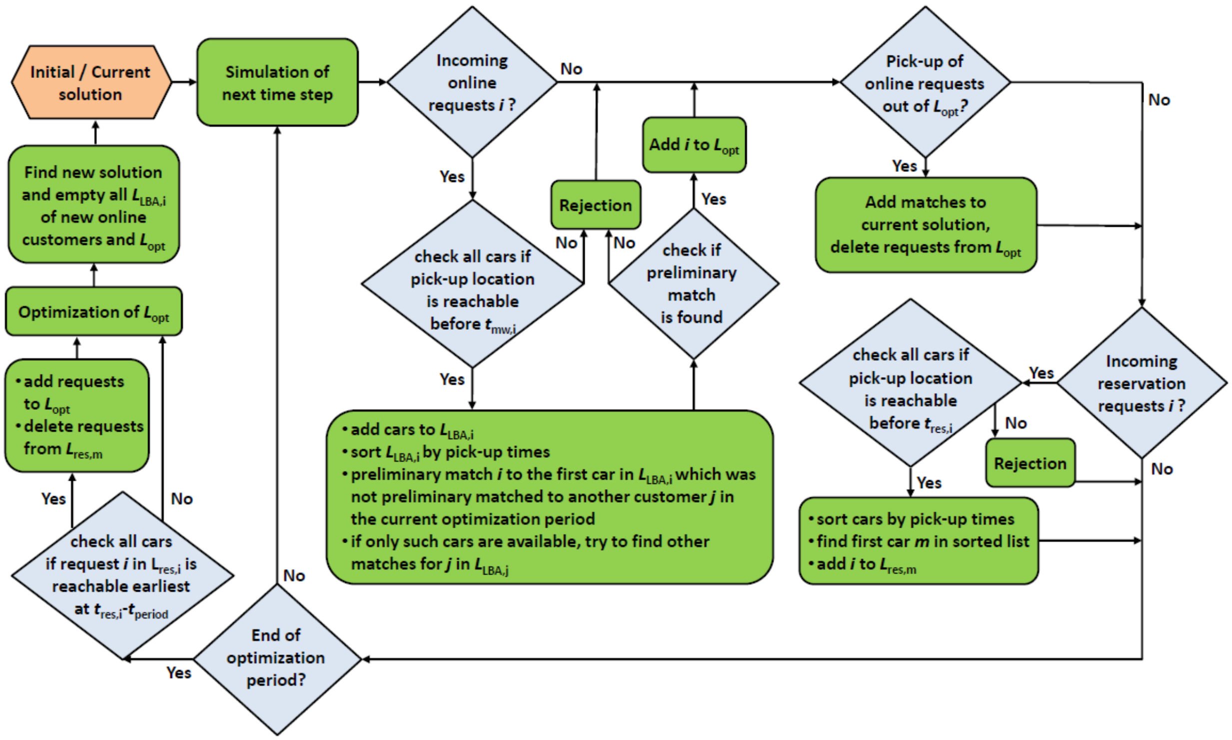


FIGURE 2 Concept of GOTW with LBAs, considering both online requests and reservations.

are not moving to the pick-up location of the next request on the list in the model proposed in this work. Instead, at the end of each optimization period with length t_{period} the list of reservations $L_{\text{res},m}$ is checked for every car $m \in M$: if a car m has to start moving to the pick-up location of its next reservation request i before the next optimization period in order to arrive there in time, the request is deleted from $L_{\text{res},m}$ and added to the route of the car. In the proposed algorithm it becomes subject to optimization (in the benchmarking algorithm using only NNP, car m has to perform the service). If request i is scheduled to be served by another car m' after the optimization and this car is not needed to start immediately to pick up the customer in time, the request is deleted from that car's route and added to $L_{\text{res},m'}$. Therefore, car m' can potentially serve future online requests or reservations $i \in N$ with earlier reservation times $t_{\text{res},j} < t_{\text{res},i}$, as long as the pick-up of customer i at $t_{\text{res},i}$ is still guaranteed.

This procedure maximizes the time cars are able to perform other tasks before picking up customers that requested the service in advance. It also produces significantly less empty driven miles, since it allows considering the spatial distribution of cars closer to the actual pick-up time. By optimizing reservation requests only when they become urgent a given time budget for optimization can be used more efficiently since the TS can focus on the solution space involving more relevant assignments of the near future, resulting in shorter computational times.

In figure 2 the principle of the GOTW with LBAs is depicted, considering online requests as well as reservations.

CASE STUDY

In this section, the simulation settings will be explained in detail before the presentation, evaluation and discussion of the results.

Simulation Settings

In this work, the open source New York Taxi Data (13) is used to simulate the demand. Because computational time for solving the problem increases with growing problem size, only 5% of all requests are taken

TABLE 1 Summary of optimization and simulation settings

TS local moves	swaps, shifts, interchanges
I_{noBSF}	10 iterations
I_{max}	100 iterations
penalty for empty driven mileage s	1 per second
waiting time penalty factor s	3 per second
rejection penalty factor r	2
online customer maximum waiting time t_{mw}	300 seconds
operator acceptance/rejection time	immediate
optimization period	60 seconds
reservation announcement time	$10 - 60$ minutes
reservation customer maximum delay	0 seconds
reservation percentages	$0\%, 50\%, 100\%$
average number of customers per hour	1000 customers
fleet sizes	$20, 40,..., 300$ cars
warm up period	60 minutes
simulation period	60 minutes
simulation time step	1 second

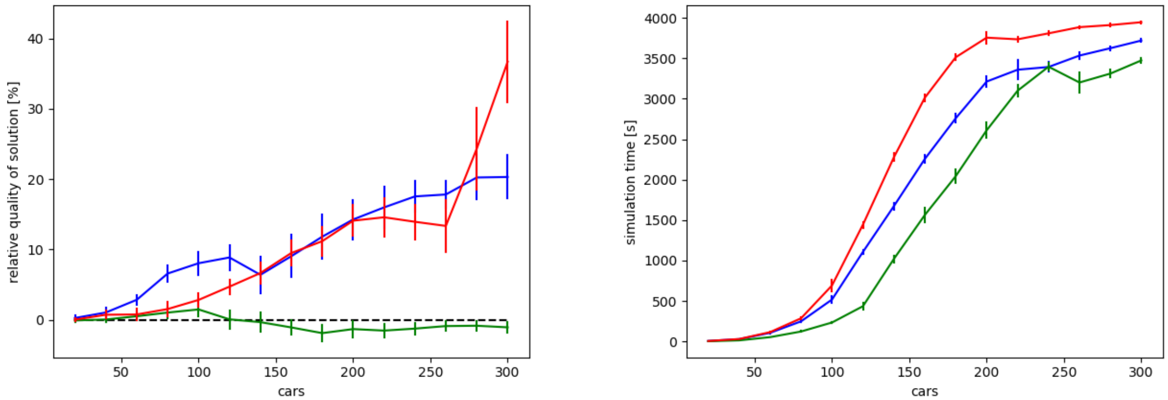
into account in this work in order to generate a meaningful number of runs to evaluate the statistical effect of the proposed methods. In each run, the selection of requests taken into account is randomly sampled. The simulations start at 6 p.m. and last until 7 p.m. of each day from Monday, June 20th 2016 to Thursday, June 23rd 2016, where around 1000 requests per simulated hour must be handled. The time of simulation is chosen to test the algorithms during the evening demand peaks in weekdays. This is the last full week the data source provides GPS information on pick-up and drop-off locations of the requests. Before each simulation, a one-hour warm up phase is run to ensure a realistic distribution of vehicles in service at the start of the simulation.

The positions of cars as well as pick-up and drop-off locations are projected on a simplified grid, measuring distances between two points on it by adding horizontal and vertical differences in position (Manhattan distance). The grid space is set to ten meters and vehicles in the simulation are able to move this distance in one simulation step which is equivalent to one second of simulated time. This leads to a constant simulated velocity of 36 kilometers per hour.

To compare the results of the proposed algorithm and the NNP, each simulation is initialized twice with the same set of cars and customers, one following the NNP only, as described in section *Nearest Neighbor Policy*, the other one using the methods explained in sections *Tabu Search*, *Global Optimization with Time Windows*, *List-Based Assignments* and *Optimization of Reservations* to find optimized solutions at each optimization step. To find the benefits of both the list-based assignments (only for online requests, as described in section *List-Based Assignments*) and the incorporation of reservations (only for scheduled requests, as described in section *Optimization of Reservations*), simulations are run using three reservation percentages: 0%, 50% and 100%.

The optimization periods are 60 seconds long which is considered as a reasonable amount of time a customer is willing to wait for the exact information when he or she will be picked up. The same applies for the maximum waiting time of online customers, which is set to 300 seconds. Reservations can be made 10 to 60 minutes in advance. Simulations including reservations last until every request becomes permanently matched and therefore contributes to the objective function value. The key performance indicators (KPIs) are measured at the end of each one hour simulation time window.

The number of available ODM fleet vehicles varies from 20 to 300 in 15 isometric steps to evaluate



(a) Qualities of solutions found with GOTW compared to NNP, representing relative values of the objective function in percent.

(b) Computational times for Tabu Search optimization in seconds.

FIGURE 3 Qualities of solutions and computational times for optimization for scenarios with 0% (green), 50% (blue) and 100% (red) reservations using GOTW (solid) and NNP (dashed).

the behavior of the algorithms with different ratios of demand (number of requests) and supply (number of cars).

Each set up is simulated ten times for each of the chosen simulation dates, leading to a total of 40 runs per setting and an overall number of 3600 simulations taken into account in this work. These simulations are run on a twelve-core Intel Xeon E5-2687W 3.0GHz processor. An overview of the scenarios is given in table 1.

In addition to that evaluation, to quantify the benefit of LBAs in terms of computational times, two sets of simulations with the same settings are initialized without the termination criterion (d) in the Tabu Search, that terminates the search after the optimization period's length. One set of simulations uses LBAs to cut the solution space at each optimization step, the other set searches the solution space defined by all cars of the fleet. The simulations consider scenarios with 50% of all requests being reservations.

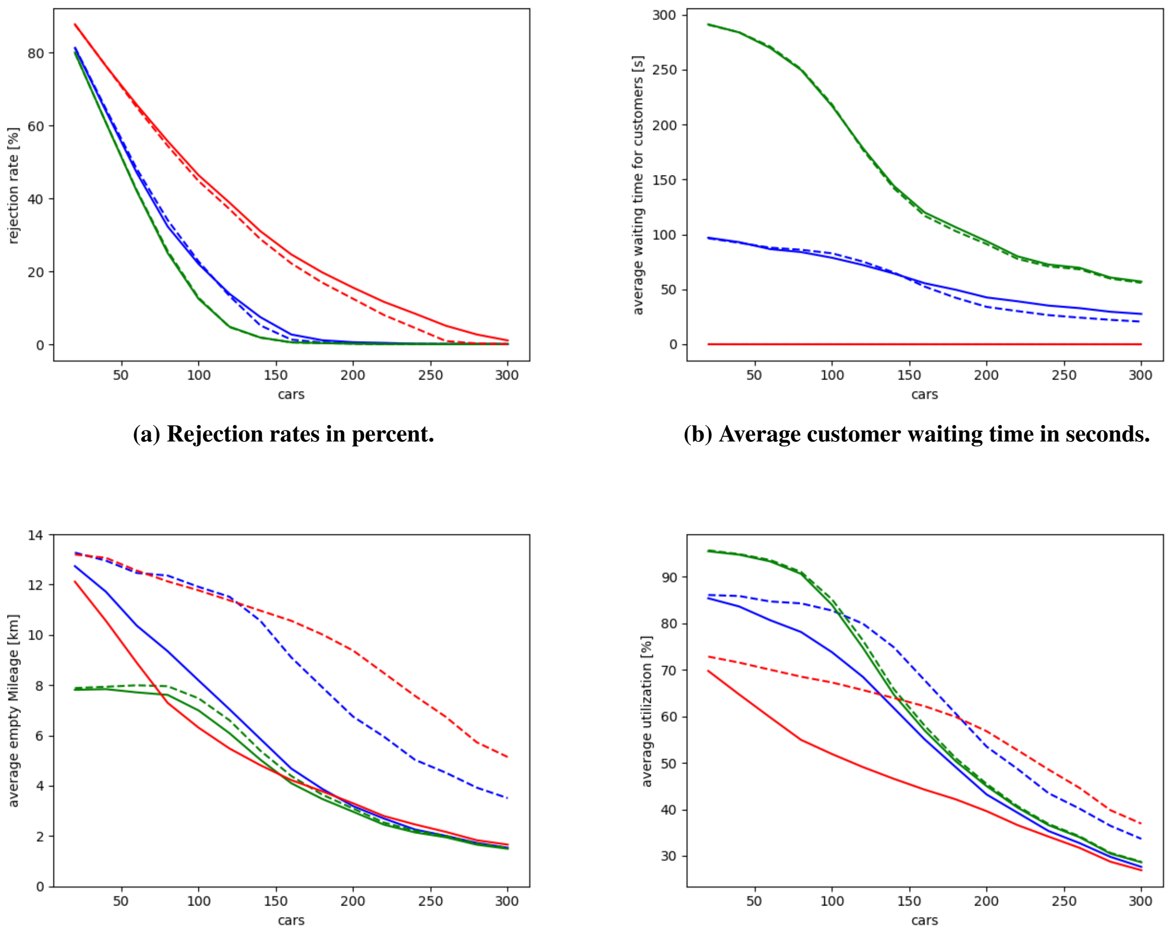
Results

The solution qualities as well as the computational times for the simulations using optimization are presented in figure 3 to evaluate the overall performances of both methods. Error bars indicate the 95%-confidence interval of the respective measurement. The quality of a solution (QoS) found with GOTW relative to solutions generated with NNP is defined as

$$\text{QoS} = \begin{cases} 1 - \frac{f_{\text{obj,GOTW}}}{f_{\text{obj,NNP}}} & \text{if } f_{\text{obj,GOTW}} > f_{\text{obj,NNP}} \\ \frac{f_{\text{obj,NNP}}}{f_{\text{obj,GOTW}}} - 1 & \text{otherwise.} \end{cases} \tag{5}$$

Additionally the average values for rejection percentages, average waiting times of customers, total empty mileages per car and fleet utilization are compared for both approaches, as shown in figure 4. To avoid confusion in the graphs, error bars are hidden in this figure.

In the case of no reservations, the overall impact of the new approach is the smallest for all ratios of available cars and customers. The rejection percentages are very similar over all simulated numbers of cars and fall below one percent in scenarios with more than 140 cars. The differences in driven and empty mileages are small in general, but the GOTW algorithm produces less empty mileage in every setting. In



(a) Rejection rates in percent.

(b) Average customer waiting time in seconds.

(c) Empty driven mileage in kilometers.

(d) Fleet utilization in percent.

FIGURE 4 KPIs for scenarios with 0% (green), 50% (blue) and 100% (red) reservations using GOTW (solid) and NNP (dashed).

scenarios with fleet sizes short of meeting the demand, the difference in produced empty mileage peaks, leading to slightly better overall solutions. With 100 cars, GOTW produces solution qualities which are 1.49% higher than those generated with NNP. On the other hand, GOTW is outperformed by the NNP in scenarios in which the rejection percentage is smaller than one percent. The worst performance is measured with 180 cars, where GOTW is 1.87% worse than NNP on average. This result is caused by the difference of around 2 seconds in average waiting time per accepted customer, which outweighs the slight benefit of saved empty mileage for fleet sizes sufficient to hold rejection rates under one percent.

When reservations are considered though, the overall solution qualities produced with GOTW are generally higher than solutions generated with NNP. If only reservations are requested (100%-reservation case), the objective function value is the sum of the penalties induced by rejections and empty miles. Each customer is either picked up at the requested pick-up time or is rejected and therefore produces no penalty due to individual waiting time. Compared to NNP, solutions found with GOTW imply generally less driven mileage per car and specifically less empty mileage. With increasing number of vehicles in the fleet, the empty mileage produced with solutions found by GOTW decreases much faster than in NNP-solutions, up to the maximum simulated fleet sizes with 300 cars. In these scenarios GOTW produces 1.66 km average empty mileage, compared to 5.14 km with NNP, a relative improvement of 67.7%. At the same time, the rejection rate in simulations using the GOTW approach is considerably higher compared to the NNP method if only reservations are considered. This behavior becomes more and more obvious with increasing fleet sizes up to the point when the demand is met, in the case of GOTW at around 300 vehicles.The empty mileage becomes increasingly dominant with more cars in the fleet, because the number of rejections decreases. Since the difference in empty mileage produced in the GOTW and NNP solutions also increases, the results found with GOTW become more and more superior to the benchmark up to maximum fleet size of 300 cars with an improvement of 36.69%. A local minimum in the solution quality is found only when the difference in rejections peaks at around 260 cars, where in solutions with NNP around 0.91% of all requests had been rejected, while with GOTW 5.11% of the customers were rejected.

In the mixed case with one half of all requests reservations, the other half customers which want to be picked up as soon as possible, the behaviors of both aforementioned cases are recognizable in the resulting KPIs. The produced empty mileage becomes more dominant with increasing number of cars, as it is the case when only reservations are considered. As the GOTW approach produces solutions with less empty mileage than in NNP's solutions and the difference in this aspect grows with increasing fleet size, the overall solutions found with GOTW tend to become better the more cars are part of the fleet. Also, this trend is only retarded when the difference in rejections reaches its peak shortly before the demand is met, as it was observed before in the 100%-reservation case. Though, the number of cars necessary to meet the demand is much more similar to the one found in the use case considering only online requests (approximately 140), with around 180 vehicles instead of between 260 and 300. The differences in average waiting times of online customers are even higher than in the 0%-reservation case for fleet sizes meeting the demand. Still, these differences are not as critical for the objective function values as the empty mileage is the dominant term in this genre. In scenarios with fleet sizes not matching the demand, the average customer waiting times produced in solutions with GOTW tend to be shorter than in those found with NNP.

The computational times for simulations rise exponentially as expected due to the kind of the optimization problem. The converging behavior for bigger fleet sizes is caused by the termination condition (d) of the Tabu Search (see section *Tabu Search*) which terminates the search after a wall-clock time equal to the optimization period's length. The higher the reservation percentage of an evaluated set up at a certain fleet size, the longer the computational time becomes. That behavior can be explained by the method of taking reservations into account in optimization: requests might be part of optimization more often than once until they are matched permanently. Online requests on the other hand are matched just once and thereafter are not part of optimization, leading to lower average numbers of requests per optimization and therefore shorter computational times in total.

In figure 5, the computational times taken for the sets of simulations using Tabu Search optimization not limited by the termination criterion (d) are compared. It is apparent that using LBAs reduces the computational effort significantly. With rising fleet size and therefore more solutions to evaluate in the optimization, the total difference in computational time taken to find solutions increases. For same fleet sizes and settings, solutions produced with LBAs are found between 7.5% (280 cars) and 31.4% (60 cars) faster compared to the equivalent algorithm without LBAs, with an average of 20.0%.

Discussion and Future Work

The results shown in the previous section give indications for strengths and weaknesses of the proposed algorithm.

A clear disadvantage is the performance in scenarios without reservations. In these scenarios, the proposed LBAs were expected to improve the solution quality by increasing the number of accepted requests compared to the simple NNP algorithm. However, due to longer average waiting times for customers the objective function values produced with LBAs and periodic optimization are worse than those found with NNP. One explanation for this behavior is the nature of the dynamic DaRP. At each optimization step, the goal of the optimization algorithm is to find the best car-passengers matches at this point in time. The

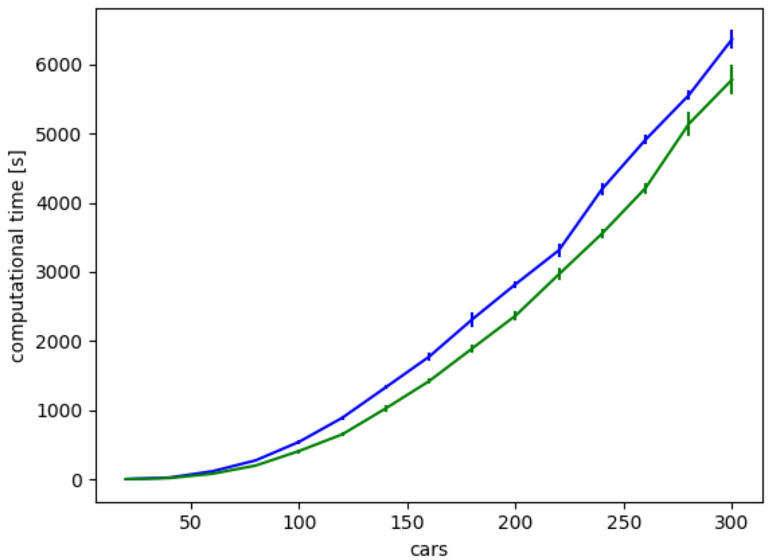


FIGURE 5 Computational times without time-out termination criterion in TS with (green) and without (blue) LBAs.

future vehicle distribution changes according to origin-destination patterns of served demand and the made assignments. While an optimal assignment at one point in time is clearly beneficial for the currently known requests, the assignments might be worse considering not yet revealed future requests. That behavior was described in detail by Dandl et al. in (15).

A similar explanation holds for the higher number of rejections in the cases with reservations. In these scenarios, cars are preliminary matched with requests for reservations as long as the cars do not need to start moving to pick up the customer in time. In that case, an optimization takes place which might find another car which is closer to the pick-up location. This procedure is repeated until no closer car is found. Then the match becomes permanent and the car starts moving to the pick-up location. As for the online scenario, the results can be explained by the dynamic problem combined with the centralized distribution of requests in the evaluated data. Cars which are able to pick up customers faster than other cars are more often close to the center, since that is where the most new requests occur. That means that those cars tend to be matched with more and more reservation requests until they do not have any capacity left to accept new ones. The NNP does not search for closer cars when a car needs to start moving in order to pick up a customer in time. This leads to much longer ways to the pick-up locations - represented by the significantly higher empty mileages in these solutions - but it also decreases the density of requests in the schedule of cars near the center which are therefore able to serve new requests more often.

The benefits of the proposed methods are clearly found in scenarios considering reservations. Especially if the fleet size is chosen to be sufficient to meet the demand and the rejection rate is very low, the approach presented in this work outperforms the NNP due to the empty mileage saved.

Using LBAs in order to avoid considering infeasible solutions in the optimization causes a significant improvement in terms of computational run time necessary to find solutions with the Tabu Search metaheuristic. This advantage allows more iterations per optimization period, potentially leading to better solutions and decreases the time necessary to find these solutions, resulting in shorter response times to customers, a crucial feature in highly dynamic problems.

Future work should further enhance the degree of realism increasing the problem scale to the actual demand and the actual sizes of ODM fleets in cities like New York City as well as using real maps, traffic information and routing instead of a simplified grid. Another emphasis could be the improvement of the search procedure itself to find better solutions in deeper solution spaces as well as the combination of the proposed model with a repositioning algorithm, which would increase the solution qualities by reducing the number of rejections and shortening individual customer waiting times.

CONCLUSION

This work's objective was to introduce an approach which is able to combine online requests of customers, which want to be picked up as soon as possible by a vehicle of an ODM fleet and reservation requests, in which customers demand a pick-up at a certain point in the future by cars of the same fleet. The proposed method is based on the concept of Global Optimization with Time Windows (GOTW). It combines the benefits of a quick initial decision, determining whether a request is accepted or rejected by a simple insertion heuristic based on the Nearest Neighbor Policy (NNP), and the optimization potential of a Tabu Search metaheuristic, which optimizes the matching of new requests and vehicles in the fleet periodically.

The presented methodology of List-Based Assignments (LBAs) is able to reduce the computational time of the Tabu Search procedure by 20% on average due to exclusion of infeasible solutions before the actual optimization takes place.

The proposed method to take requests for reservation into account in the optimization of the car-passenger matches generates less empty mileage driven by the ODM fleet than the NNP (up to 67.7% in scenarios with 300 cars and 100% reservations). This benefit outweighs the higher rejection rates in solutions found with the proposed algorithm compared to the benchmark solutions following the NNP by up to 36.69%.

Upcoming work should evaluate the shortcomings of the presented approaches in scenarios with no reservations, where the average customer waiting time is higher than in the benchmark solutions. Also, repositioning of idle cars to decrease average waiting times and rejection rates should be considered.

AUTHOR CONTRIBUTION

Marvin Erdmann, Florian Dandl, Bernd Kaltenhäuser and Klaus Bogenberger designed the research; Marvin Erdmann implemented the simulations and data analysis. Marvin Erdmann, Florian Dandl and Bernd Kaltenhäuser interpreted the results and prepared the manuscript. All authors reviewed the results and approved the final version of the manuscript.

References

[1] Psaraftis, H. N. A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, 1980. 14: 130–154.

[2] Osman, I. H., and J. P. Kelly. Meta-Heuristics: An Overview. In *Meta-Heuristics* (I. H. Osman and J. P. Kelly, eds.), Springer US, Boston, 1996, pp. 1–21.

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

[4] Brandão, J. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research*, 2011. 38: 140–151.

[5] Prodhon, C., and C. Prins. Metaheuristics for Vehicle Routing Problems. In *Metaheuristics* (P. Siarry, ed.), Springer International Publishing Switzerland, Basel, 2016, pp. 407–437.

[6] Pandi, R. R., S. G. Ho, S. C. Nagavarapu, T. Tripathy, and J. Dauwels. GPU-Accelerated Tabu Search Algorithm for Dial-A-Ride Problem. *Proceedings of 21st International IEEE Conference on Intelligent Transportation Systems*, 2018. 2519–2524.

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

[8] Hyland, M., and H. S. Mahmassani. Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests. *Transportation Research Part C*, 2018. 92: 278–297.

[9] Sheridan, P. K., E. Gluck, Q. Guan, T. Pickles, B. Balcıoḡlu, and B. Benhabib. The Dynamic Nearest Neighbor Policy for the Multi-Vehicle Pick-Up and Delivery Problem. *Transportation Research Part A*, 2013. 49: 178–194.

[10] Uber. *Riding with Uber - Schedule a Ride*. Uber Technologies Inc., San Francisco. https://www.uber.com/ride/how-uber-works/scheduled-rides. Accessed June 26, 2019.

[11] Lyft. *Scheduled Rides for Passenger*. Lyft Inc., San Francisco. https://help.lyft.com/hc/en-us/articles/115013078668-Scheduled-rides-for-passengers. Accessed June 26, 2019.

[12] Erdmann, M., F. Dandl, and K. Bogenberger. Dynamic Car-Passenger Matching based on Tabu Search using Global Optimization with Time Windows. Presented at the 8th International Conference on Modeling, Simulation and Applied Optimization, Manama, Bahrain, 2019.

[13] Taxicab Passenger Enhancement Program. *2016 Yellow Taxi Trip Data*. NYC Taxi and Limousine Commision, New York City. https://data.cityofnewyork.us/Transportation/2016-Yellow-Taxi-Trip-Data/k67s-dv2t. Accessed May 15, 2019.

[14] Dandl, F., and K. Bogenberger. Booking Processes in Autonomous Carsharing and Taxi Systems. *Proceedings of 7th Transportation Research Arena*, Vienna, Austria, 2018.

[15] Dandl, F., M. Hyland, K. Bogenberger, and H. S. Mahmassani. Evaluating the impact of spatio-temporal demand forecast aggregation on the operational performance of shared autonomous mobility fleets. *Transportation*, 2019. 114: 462–484