A New Lower Bound for the Static Dial-a-Ride Problem with Ride and Waiting Time Minimization

Christian Pfeiffer(✉) and Arne Schulz

Institute of Operations Management, University of Hamburg, Hamburg, Germany
{Christian.Pfeiffer,Arne.Schulz}@uni-hamburg.de

Abstract. The paper focuses on the static dial-a-ride problem with ride and waiting time minimization. This is an important problem setting of significant practical relevance, as several ridesharing providers launched in recent years in large cities. In contrast to the standard dial-a-ride problem, these providers focus on the general public. Therefore, they are amongst others in competition with taxis and private cars, which makes a more customer-oriented objective necessary. We minimize the sum of relative detours of all customers. The paper introduces upper bounds for the arrival times and an initial lower bound for the objective value. Our approach is tested in a computational study with realistic test instances.

Keywords: Demand responsive transport · Dial-a-Ride · Branch-and-bound · Lower bound

1 Introduction

Private mobility in cities grows more and more important. In many cities, the residents have access to a wide range of motorized mobility options including public transportation, taxis or their private car. Over the past years ridesharing has grown to be a competitive alternative to the previous options. Uber, the biggest company in that area, connects private drivers and potential customers. MOIA—a subsidiary company of the automobile manufacturer Volkswagen—, ioki or CleverShuttle—subsidiary companies of the main German railway company Deutsche Bahn—have different business models to Uber [1,9,12]. They use their own fleet of cars and employed drivers. MOIA, ioki, and CleverShuttle use apps where customers can ask for a ride from a pickup location to a delivery location. The app matches the requested rides and makes an offer the customer can accept or decline. As the prices are lower than those for taxis, this concept works only if customer requests are combined as efficiently as possible. On the one hand, the customers decline tours that have a too long detour and use public transport, their private car or a taxi instead. On the other hand, few shared rides are not cost-effective for the provider.

This paper develops a new lower bound for the optimization problem a provider of such a ridesharing service encounters. The problem is known as the dial-a-ride problem (DARP) in the literature and often optimized to minimize costs. In this paper, we consider an objective function for the DARP that does not aim to minimize costs but instead minimizes the customer inconvenience and was first introduced in Pfeiffer and Schulz [19]. Minimizing customer inconvenience allows the service provider to compete better with other modes of transport.

The paper is structured as follows: First, we classify our problem setting into the literature (Sect. 2) and give a detailed problem description (Sect. 3). Then, we present the lower bound in Sect. 4. This is followed by a computational study in Sect. 5. Afterwards, we feature a conclusion in Sect. 6.

2 Literature Review

While customer inconvenience is often included in traditional formulations of the DARP [16], it is often modeled with the help of constraints: time windows and maximum ride time restrictions are usually utilized [2,6,21]. Our formulation forgoes these constraints and relies on the objective function to minimize the customer inconvenience.

The DARP was originally proposed by Psaraftis [20], who solves the problem with dynamic programming. Since then there has been a large variety of employed methods to solve the DARP. Cordeau [2] and Ropke et al. [21] use Branch-and-Cut algorithms to solve the DARP exactly. One of the models presented in Ropke et al. [21] is the basis for the model formulation presented in this paper. Gschwind and Irnich [6] and Parragh et al. [18] utilize Branch-and-Price methods. For their Branch-and-Price algorithm, Gschwind and Irnich [6] split the original problem into a master and a subproblem. The master problem selects the tours while the subproblem aims to identify tours that improve the solution. The subproblem is solved with a labeling algorithm that utilizes dynamic time windows to reduce the size.

Pfeiffer and Schulz [19] first introduced the DARP variant that we examine in this paper. They present an Adaptive Large Neighborhood Search and a Dynamic Program to solve the problem. The Dynamic Program can only solve instances with a single vehicle while the Adaptive Large Neighborhood Search algorithm can be utilized for multi-tour instances as well.

Much of the literature for the DARP considers the transport of elderly, ill or handicapped passengers [4,7,11]. Nevertheless, the DARP has also been investigated in the context of demand responsive transport in urban areas [10,18]. For more in-depth reviews of the DARP literature we refer to Ho et al. [8] and Parragh et al. [16].

A brief overview about papers that minimize customer inconvenience in a DARP is presented in Table 1. The majority of the examined literature considers some form of cost minimization objective as well.

Table 1. DARP literature about minimizing customer inconvenience ALNS: Adaptive Large Neighborhood Search DP: Dynamic Program GA: Genetic Algorithm MDLS: Multi-Directional Local Search PR: Path Relinking TS: Tabu Search VND: Variable Neighborhood Descent VNS: Variable Neighborhood Search

Authors (year)	Objective	Constraints	Algorithm
Sexton and Bodin (1985) [22]	Min. excess ride time, min. delivery time deviation	Vehicle capacity, latest delivery time	Benders
Diana and Dessouky (2004) [5]	Min. total distance, min. excess ride time, min. idle time	Max. ride time, vehicle capacity, time windows	Regret insertion
Jorgensen et al. (2007) [11]	Min. total transportation time, min. excess ride time, min. wait time	Vehicle capacity, soft time windows	GA
Parragh et al. (2009) [17]	Min. total distance, min. mean user ride time	Max. ride time, vehicle capacity, time windows	VNS, PR
Paquette et al. (2013) [15]	Min. total distance, min. mean (relative) user ride time, min. time, window deviations	Max. ride time, vehicle capacity, time windows, vehicle incompatibilities	TS
Molenbruch et al. (2017) [13]	Min. total distance, min. mean user ride time	Max. ride time, vehicle capacity, time windows, pairing restrictions	MDLS, VND
Pfeiffer and Schulz (2021) [19]	Min. relative ride and waiting time	Vehicle capacity	ALNS, DP

3 Problem Description

Compared to the standard DARP formulation [3], our problem features differences in the objective function as well as some of the constraints (compare [19]). Similar to the standard formulation we consider a number of requests n, each with their own given demand (i.e. the number of passengers in the request). There are K homogeneous vehicles, which all start and end at the same depot, and all have the same capacity Q. For each request $i \in P$ there is a pickup ($P = \{1, ..., n\}$) with demand $q_i > 0$ and a delivery location $i + n \in D$ ($D = \{n + 1, ..., 2n\}$) with demand $q_{i+n} = -q_i$ which need to be visited by the same vehicle and in the order pickup before delivery. Request $i \in P$ has the earliest pickup time r_i. Between all pairs of locations there are deterministic travel times t_{ij} which fulfill the triangle inequality. Accordingly, $r_{i+n} = r_i + t_{i,i+n}$ for all $i \in P$. If a vehicle arrives at a customer's pickup location before the request time, it has to wait. Every customer needs to be serviced. We use distances and travel times synonymously in the following.

Unlike the standard DARP formulation we do not minimize the driven distances. Instead we minimize the customer inconvenience by minimizing the sum of relative detours over all customers. The detour is defined as the sum of waiting and riding times. More formally, the detour is calculated by subtracting the sum of the request time and the direct travel time between pickup and delivery location from the arrival time at the delivery location. The detour is then divided by the direct travel time between pickup and delivery location to receive the relative detour. It is weighted by the number of passengers q_i of request $i \in P$. By measuring the objective this way, a given nominal detour will have a larger effect on a customer with a short direct route than on one with a longer direct route. This reflects the actual inconvenience that many customers perceive in such a situation [14].

In metropolitan areas, ridesharing providers compete with several other modes of transportation such as public transport, taxis or private cars. As a result, it is paramount for the ridesharing company to be attractive for its customer base. Since many ridesharing providers utilize electric cars, reducing the traveled distance is furthermore of reduced importance, as recharging is cheap. In addition, should autonomous vehicles become more commonplace, driver costs can further be disregarded which reinforces the need to focus on customer inconvenience.

Due to the minimization of customer inconvenience in the objective function, we omit maximum ride time restrictions and time windows. As a result, there is always a trivial feasible solution for our problem as long as no customer demand is higher than the vehicle's capacity. Simply sequencing each delivery location right after its pickup location will never violate the capacity constraint in that case. The journey time of each customer will also be minimized in such a solution, but there might be large waiting times. Our objective will therefore aim to pool different requests together, which increases the journey time in favour of the waiting time, to reduce the overall customer inconvenience.

4 Mixed-Integer Program

In this section, we present a 2-index formulation for our problem setting (compare [19]). We define the problem on a directed graph $G = (I, A)$, where I is the set of nodes and A is the set of arcs. Let $I = \{0, 2n+1\} \cup P \cup D$ be the set of all locations, where 0 and $2n + 1$ represent the depot with $r_0 = r_{2n+1} = 0$ and $q_0 = q_{2n+1} = 0$. We use the indices $i, j \in I$. Moreover, let M_{ij} be sufficiently large constants. We use the following variables: $X_{ij} = 1$ if a vehicle drives from location i directly to location j and 0 otherwise, Q_i specifies the number of passengers in the vehicle after leaving location i, and B_i indicates the departure time from location i.

We assume that a request time r_i is never smaller than the distance from the depot to the pickup node i, i.e. $r_i \geq t_{0,i}$ $\forall i \in P$, which means that visiting only a single customer in a tour will always result in an objective value of 0 for that tour.

4.1 Model Formulation

We need to define the sets $S \subseteq I$ and $\mathcal{S}$. $\mathcal{S}$ is the set of all sets S for which $0 \in S$, $2n+1 \notin S$ and there is at least one request i that has its delivery node in S but not the pickup node, i.e., $\mathcal{S} = \{S : 0 \in S \wedge 2n+1 \notin S \wedge \exists i : (i \notin S \wedge n+i \in S)\}$. The model formulation is then:

$$\min \quad \sum_{i \in P} q_i \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} \tag{1}$$

$$\sum_{i \in I} X_{ij} = 1 \qquad \forall j \in P \cup D \tag{2}$$

$$\sum_{j \in I} X_{ij} = 1 \qquad \forall i \in P \cup D \tag{3}$$

$$\sum_{j \in P} X_{0j} = K \tag{4}$$

$$\sum_{i,j \in S} X_{ij} \leq |S| - 2 \qquad \forall S \in \mathcal{S} \tag{5}$$

$$B_i + t_{ij} - M_{ij}(1 - X_{ij}) \leq B_j \qquad \forall i \in I, j \in P \cup D \tag{6}$$

$$Q_i + q_j - Q(1 - X_{ij}) \leq Q_j \qquad \forall i \in I, j \in P \cup D \tag{7}$$

$$B_i \geq r_i \qquad \forall i \in I \tag{8}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \qquad \forall i \in I \tag{9}$$

$$X_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in I \tag{10}$$

The model was previously introduced in Pfeiffer and Schulz [19] and is based on the model in Ropke et al. [21]. Compared to Pfeiffer and Schulz [19] we set the discussed penalty factor for maxima w to 0 and therefore omit it from our model formulation, as this leads to a fair evaluation of our lower bound. A verbal description for the remaining model can be found in Pfeiffer and Schulz [19]. To find violations of Constraints (5), we use the same procedure as Ropke et al. [21]: Each time an integer solution is found in the Branch-and-Bound tree, we solve a maximum flow problem for every customer to determine whether the precedence constraints are met. If they are violated, the corresponding cuts are added and the solution is marked as infeasible.

4.2 Calculation of an Initial Lower Bound

In order to reduce the initial gap, we provide a lower bound for the objective value to the solver. A slightly modified version of this lower bound can also be used to tighten the bounds of the B_i variables (compare Subsect. 4.3). In order to calculate the lower bound, we solve subproblems of our original problem by only considering a subset of customers. We generate all possible subsets of customers with sizes $[K+1, \dots, K+\phi]$ with $\phi \geq 1$. Each subproblem is solved to optimality and the corresponding objective value is saved. We do not consider smaller subsets because for an instance with $n \leq K$ the optimal objective value is always 0. The subsets are solved to optimality by going through all possible assignments of customers to tours and then solving the individual tours with the help of the dynamic program in Pfeiffer and Schulz [19]. The objective value for each tour is stored in a hash table and reused throughout the procedure to decrease the running time. These individual subproblems each constitute a lower bound for the original instance. However, combining them improves the bound further: We look for a combination of subproblems where no customer appears more than once across the combined subproblems. The maximum combination, i.e. the combination where the sum of the respective objective values is maximal, is then a lower bound for the original problem. This selection problem is solved with CPLEX.

When evaluating a subproblem l with the customers $P_l \subset P$, we also utilize the already calculated solutions for all smaller subsets m with customers $P_m \subset P_l$ to further reduce the computational effort. The largest optimal objective value of these smaller subsets m constitutes a lower bound for the objective of the larger subset l since adding a customer can never improve the objective value. If a particular assignment of customers to tours for subset l returns an objective value equaling the largest lower bound, the rest of the possible assignments do not need to be considered.

We use this procedure also to generate n additional lower bounds where we remove a single customer each. This is accomplished by only combining subproblems where the customer does not appear. These additional n lower bounds are used in Subsect. 4.3 to improve the bounds for B_i.

4.3 Bounding of Variables

In order to tighten the variable bounds for the model formulation, we use the objective value of a feasible initial solution and the individual lower bounds for the problem without customer i from Subsect. 4.2. Let Z^{UB} be the objective value of an initial solution and Z_i^{LB} the lower bound for the problem without customer i. Let B_i^{LB} and B_i^{UB} denote the lower and upper bound of variable B_i. We set them as follows:

$$B_i^{LB} = \begin{cases} r_i & \forall i \in P \\ r_i + t_{i-n,i} & \forall i \in D \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

and

$$B_i^{UB} = \begin{cases} \frac{(Z^{UB} - Z_i^{LB}) \cdot t_{i,i+n}}{q_i} + r_i & \forall i \in P \\ B_{i-n}^{UB} + t_{i-n,i} & \forall i \in D \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

While the lower bounds in (11) are already included in the model formulation (1)–(10) by (8) and the restriction that $i + n$ is visited after i by the same vehicle, the idea behind the upper bounds in (12) is that we already know that in the optimal solution, B_i of a single customer i can never be so large that the customer i would generate an objective value greater than that of the initial solution. This bound can be improved if we know how much of the objective value all the other customers will generate at minimum, which is the lower bound for the instance without the customer i. The upper bounds in (12) have, to the best of our knowledge, never been used before.

Furthermore, B_i^{LB} and B_i^{UB} allow us to set

$$M_{ij} = B_i^{UB} - B_j^{LB} + t_{ij} \quad \forall i \in I, j \in P \cup D.$$

where applicable, we also adapted the bound tightening constraints from Cordeau [2] and Ropke et al. [21] for the variables B_i and Q_i.

5 Computational Study

The Branch-and-Bound algorithm was implemented in C++, using the CPLEX C++ API and CPLEX version 12.9. All tests were run on a single thread of an AMD Ryzen 3990X and with 256 GB of RAM.

5.1 Instances

The instances are generated based on a district in the city of Hamburg, Germany, where a dial-a-ride service is operating. Within the district there are virtual stops which are used by the dial-a-ride provider. We draw the request locations randomly from these virtual stops.

The request times are uniformly distributed but will always be large enough for a vehicle to drive straight from the depot to a pickup location and arrive before or at the request time. More precisely, if $maxT = \max_{i \in P} t_{0i}$ is the maximum distance from the depot to a pickup location, the request times are drawn from the range $[maxT, maxT + \beta \cdot n]$ with $\beta \geq 0$. β controls the temporal concentration of the requests and was set to 100 in our experiments, as this yielded instances that were challenging but not too difficult. Higher values for β tend to make the instances easier, as there is more time between requests on average. For smaller values of β the system's capacity is exceeded as requests arrive faster than they can be completed. The distances between the locations are not symmetrical as the city layout is taken into account. However, all distances fulfill the triangle inequality.

The vehicle capacity was set to 6 passengers, which is common in practice. The demand of each request (i.e. the number of passengers) is determined by an exponential distribution with a λ of 0.9. The value is rounded up. If the resulting value is not in the interval $[1, 6]$, the value is discarded and redrawn until it is in the valid range.

The number of requests n was varied between 10, 15, 20, and 30. We varied the number of tours K between 1, 2, and 3 for all three n. We generated 10 instances for every combination, which results in a test bed of 120 instances in total.

For the mixed-integer program (MIP) we set the CPLEX parameter *MIP emphasis* to *bestbound* which helps in reducing the optimality gap. The other CPLEX settings are set to their defaults unless otherwise noted.

5.2 Discussion of Results

We performed tests for several different configurations: the MIP without a starting solution and without the preprocessing lower bound procedure (MIP_0), and the MIP with the starting solution mentioned in Subsect. 4.3 (generated by the Adaptive Large Neigborhood Search (ALNS) in Pfeiffer and Schulz [19]) and with the preprocessing lower bound using different values for the parameter ϕ as described in Subsect. 4.2 (MIP_ϕ). ϕ is varied between 1, 2, 3, and 4. Finally, we ran an extra MIP configuration where we used depth first search and set the CPLEX parameter *MIP emphasis* to *feasibility* (MIP_4^{DF}). This last configuration is otherwise identical to MIP_4 and was used to put more focus on improving the upper bound. With the exception of MIP_0, all other configurations used the improved bounds from (12). Each CPLEX run had an hour of computation time which was exhausted for almost all instances. Only some of the smaller instances with $n = 10$ could be solved to optimality. MIP_4^{DF} did so most often (in 10 of the 120 instances).

In Table 2, the average upper bounds for all MIP configurations as well as the starting solution by ALNS are displayed. The configuration with the best average upper bound is MIP_4^{DF} but even MIP_4^{DF} cannot improve the starting solutions much in the one hour run, which indicates that CPLEX struggles with the model formulation. MIP_0 performs worst since it does not receive the initial ALNS solution and has difficulties to find feasible solutions for instances with $n > 10$. None of the feasible results of MIP_0 ever beat the initial solution. Configurations MIP_1 to MIP_4 only vary slightly from one another and only rarely find a better solution than the initial one. MIP_1, for example, can only improve on the initial solution's upper bound in 2 out of the 120 instances.

Table 2. Average upper bounds of all MIP configurations and the intial solution from ALNS. For config MIP_0, – indicates that not all 10 instances were solved to feasibility

n	K	MIP_0	MIP_1	MIP_2	MIP_3	MIP_4	MIP_4^{DF}	ALNS
10	1	61.30	59.69	59.69	59.69	59.69	59.69	59.69
10	2	21.84	21.80	21.80	21.80	21.80	21.80	21.80
10	3	8.17	8.17	8.17	8.17	8.17	8.17	8.17
15	1	–	113.19	113.19	113.19	113.19	113.19	113.19
15	2	–	48.34	48.34	48.34	48.34	48.34	48.34
15	3	–	24.30	24.30	24.30	24.30	24.30	24.30
20	1	–	187.75	187.80	187.75	187.80	187.62	187.80
20	2	–	78.89	78.89	78.89	78.89	78.78	78.89
20	3	–	37.06	37.06	37.06	37.06	37.06	37.06
30	1	–	382.00	382.00	382.00	382.00	381.79	382.00
30	2	–	151.33	151.33	151.34	151.34	150.89	151.34
30	3	–	78.73	78.73	78.73	78.73	78.72	78.73

Table 3 displays the average lower bounds before (LB^b) and after (LB^a) the execution of CPLEX in every configuration, i.e. the lower bound after our lower bound procedure was executed and the lower bound after the CPLEX run. Since we did not execute the lower bound procedure in the configuration MIP_0, the respective "before" column is omitted (and effectively 0 for every entry). Clearly, as ϕ is increased, the lower bound is increased as well since more combinations are evaluated. For $\phi \geq 2$, the lower bound improvement procedure can clearly outperform the lower bound that is received when only running CPLEX for an hour (MIP_0). In instances with few vehicles and many customers, this effect is more pronounced. In the most extreme case, for $n = 30$ and $K = 1$ the average lower bound is improved by more than 400% (compare the result of MIP_4). Additionally, for instances with few vehicles the lower bound preprocessing is very fast as well: For $n = 30$ and $K = 1$ the average computation time was less than three minutes when setting $\phi = 4$ (compare Table 4).

The difficulty of solving the model with a traditional MIP solver is further reemphasized by the fact that the preprocessing lower bounds are rarely if ever improved in the subsequent CPLEX run where the lower bound was provided as an initial lower bound. Combining this result with the insight that the upper bounds are also only barely improved, it seems that using a heuristic method in combination with the lower bound preprocessing beats a traditional MIP solver in both solution gap as well as time.

The computation times for the preprocessing lower bound can be seen in Table 4. For configurations with low ϕ or instances with low K, the computation times are negligible. For instances with a large number of vehicles, the computa-

Table 3. Average lower bounds before (LB^b) and after (LB^a) the CPLEX run of different MIP configurations

n	K	MIP_0	MIP_1		MIP_2		MIP_3		MIP_4		MIP_4^{DF}	
		LB^a	LB^b	LB^a	LB^b	LB^a	LB^b	LB^a	LB^b	LB^a	LB^b	LB^a
10	1	21.83	14.33	17.34	23.05	23.05	30.25	30.25	38.36	38.36	38.36	38.36
10	2	13.02	5.88	11.30	8.74	11.13	12.04	12.76	13.39	13.79	13.39	14.42
10	3	7.68	2.37	7.64	3.90	7.30	4.42	7.42	5.23	7.66	5.23	7.80
15	1	16.79	21.18	21.18	35.80	35.80	45.84	45.84	56.42	56.42	56.42	56.42
15	2	11.72	9.56	10.16	14.16	14.16	19.29	19.29	22.34	22.34	22.34	22.34
15	3	9.16	5.40	8.39	8.97	9.20	10.71	10.71	13.30	13.30	13.30	13.30
20	1	19.67	29.00	29.00	48.21	48.21	64.84	64.84	78.35	78.35	78.35	78.35
20	2	12.43	12.27	12.27	18.95	18.95	25.44	25.44	30.40	30.40	30.40	30.40
20	3	9.05	6.71	7.59	10.32	10.32	12.58	12.58	15.13	15.13	15.13	15.13
30	1	21.15	44.11	44.11	74.55	74.55	96.75	96.75	119.18	119.18	119.18	119.18
30	2	15.18	18.76	18.76	28.07	28.07	36.79	36.79	44.56	44.56	44.56	44.56
30	3	11.31	10.67	10.77	16.43	16.43	20.84	20.84	25.47	25.47	25.47	25.47

tion times increase substantially when also setting $\phi = 4$. However, they are still well below the allocated time of the MIP while generating higher quality lower bounds. In future research, the computation times could be reduced if the subproblems are selected heuristically instead of evaluating all of them. The same table also shows the average relative improvement of the lower bounds when incrementing ϕ. Since the relative improvement of the lower bounds decreases with each additional increase of ϕ, lower values for ϕ can be worthwhile to employ, as they maintain computation times rarely exceeding a minute.

In Fig. 1, the occupancy rate for the various instances is visualized. Understandably, reducing the number of vehicles and increasing the number of requests increases the occupancy rate per vehicle. This increases the profit for the service provider but is unattractive for the customers. A low occupancy rate, however, is economically not sustainable for the provider but beneficial for the customers. It is therefore important to find a suitable number of vehicles that provides an adequate service level (see Table 2) while still being economically viable.

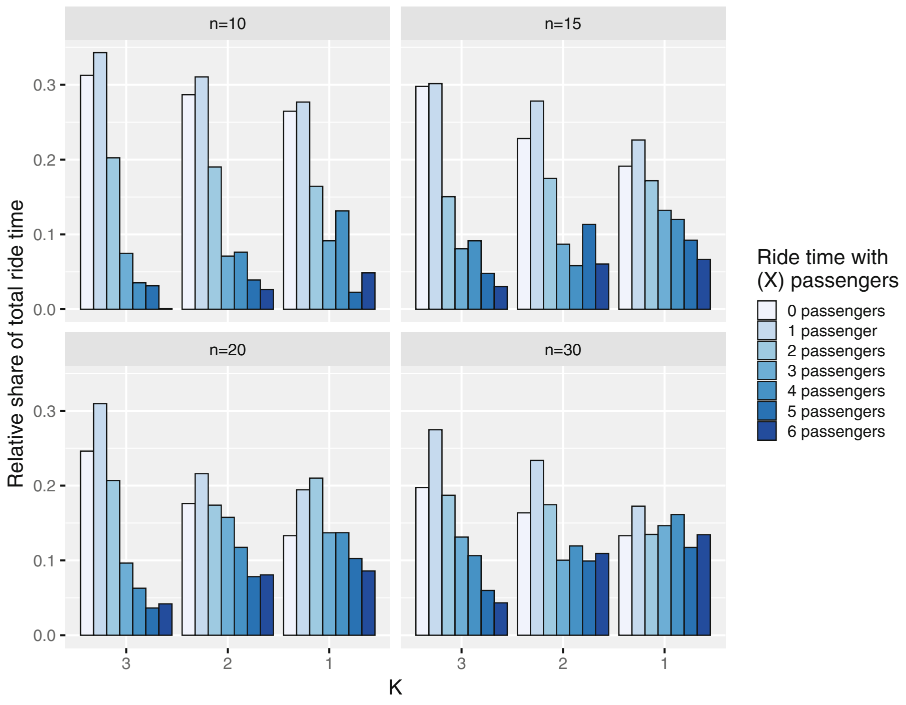

Fig. 1. Development of the occupancy rate when varying K and n for the solutions by MIP_4. Each bar is the relative share of ride time with that amount of customers. For example, for $n = 10$ and $K = 3$ the vehicles have two passengers in 20% of the time.

Table 4. Average time for the calculation of the preprocessing lower bound in seconds (t) and the average relative improvement of the lower bound (Δ_{LB}) when incrementing ϕ

n	K	t (s)				Δ_{LB} (%)		
		$\phi = 1$	$\phi = 2$	$\phi = 3$	$\phi = 4$	ϕ : 1 to 2	ϕ : 2 to 3	ϕ : 3 to 4
10	1	0.00	0.01	0.10	0.53	60.90	30.67	26.74
10	2	0.01	0.02	0.10	0.50	47.72	38.33	10.65
10	3	0.01	0.02	0.12	0.57	63.57	13.91	17.97
15	1	0.01	0.04	0.49	4.88	69.90	28.76	23.42
15	2	0.01	0.07	0.57	5.27	47.77	36.46	16.06
15	3	0.04	0.15	0.93	6.30	64.42	20.57	24.93
20	1	0.01	0.09	1.42	20.79	66.85	34.67	20.80
20	2	0.03	0.19	1.91	23.42	53.99	34.27	19.95
20	3	0.06	0.41	3.89	40.88	56.10	22.69	20.83
30	1	0.01	0.25	6.63	159.84	69.05	29.77	23.11
30	2	0.05	0.80	11.67	188.54	49.73	31.16	21.09
30	3	0.20	2.40	40.30	557.24	54.02	26.69	22.31

6 Conclusion

The paper presents a new initial lower bound for the static dial-a-ride problem with ride and waiting time minimization. The initial lower bound outperformed the lower bound of CPLEX after one hour of computation in almost all of our 120 instances. Furthermore, the initial solutions of the ALNS were rarely beaten by CPLEX after an hour of computation. Therefore, we conclude that using the initial lower bound in combination with a metaheuristic such as ALNS is more beneficial relating to computation time and solution quality than using CPLEX with one hour of computation time.

Our research leads to several possible directions for further research. First of all, the computation times for the initial lower bound can be improved by selecting the solved subproblems heuristically. In the same way, ϕ can be increased without increasing the computation time. Furthermore, the problem formulation can be adapted such that the customers can determine their latest arrival times at the delivery location instead of their earliest departure times (request times). The model determines the departure times instead of the arrival times in this case. This is a realistic problem setting because customers might have important appointments at their destination and want to be sure to be on time.

Funding. This project was supported by the Hamburger Behörde für Wissenschaft, Forschung, Gleichstellung und Bezirke (BWFGB; Hamburg authority for science, research, equalization, and districts). No grant number is available.

Availability of Data. The instances are available under the following link: http://doi.org/10.25592/uhhfdm.9670.

Conflict of Interest. The authors declare that they have no conflict of interest.

References

1. CleverShuttle: Shared rides for a better furture (2022). https://www.clevershuttle.de/en/. Accessed 03 Jan 2022
2. Cordeau, J.F.: A branch-and-cut algorithm for the dial-a-ride problem. Oper. Res. **54**(3), 573–586 (2006)
3. Cordeau, J.F., Laporte, G.: The dial-a-ride problem: models and algorithms. Ann. Oper. Res. **153**(1), 29–46 (2007)
4. Detti, P., Papalini, F., de Lara, G.Z.M.: A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega **70**, 1–14 (2017)
5. Diana, M., Dessouky, M.M.: A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transp. Res. Part B Methodol. **38**(6), 539–557 (2004)
6. Gschwind, T., Irnich, S.: Effective handling of dynamic time windows and its applications to solving the dial-a-ride problem. Transp. Sci. **49**(2), 335–354 (2015)
7. Heilporn, G., Cordeau, J.F., Laporte, G.: An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays. Discret. Appl. Math. **159**(9), 883–895 (2011)
8. Ho, S.C., Szeto, W.Y., Kuo, Y.H., Leung, J.M.Y., Petering, M., Tou, T.W.H.: A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. Part B Methodol. **111**, 395–421 (2018)
9. ioki GmbH: ioki—inspiring smart mobility (2022). https://ioki.com/en/home/. Accessed 03 Jan 2022
10. Jaw, J.J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.: A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transp. Res. Part B Methodol. **20**(3), 243–257 (1986)
11. Jorgensen, R.M., Larsen, J., Bergvinsdottir, K.B.: Solving the dial-a-ride problem using genetic algorithms. J. Oper. Res. Soc. **58**(10), 1321–1331 (2007)
12. MOIA GmbH: Reunion begins here (2022). https://www.moia.io/en. Accessed 03 Jan 2022
13. Molenbruch, Y., Braekers, K., Caris, A., den Berghe, G.V.: Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation. Comput. Oper. Res. **77**, 58–71 (2017)
14. Nie, W.: Waiting: integrating social and psychological perspectives in operations management. Omega **28**(6), 611–629 (2000)
15. Paquette, J., Cordeau, J.F., Laporte, G., Pascoal, M.M.: Combining multicriteria analysis and tabu search for dial-a-ride problems. Transp. Res. Part B Methodol. **52**, 1–16 (2013)
16. Parragh, S.N., Doerner, K.F., Hartl, R.F.: A survey on pickup and delivery models: part ii: transportation between pickup and delivery locations. J. für Betriebswirtschaft **58**(2), 81–117 (2008)
17. Parragh, S.N., Doerner, K.F., Hartl, R.F., Gandibleux, X.: A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks **54**(4), 227–242 (2009)

18. Parragh, S.N., de Sousa, J.P., Almada-Lobo, B.: The dial-a-ride problem with split requests and profits. Transp. Sci. **49**(2), 311–334 (2015)
19. Pfeiffer, C., Schulz, A.: An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization. OR Spectr. **44**, 87–119 (2022)
20. Psaraftis, H.N.: A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transp. Sci. **14**(2), 130–154 (1980)
21. Ropke, S., Cordeau, J.F., Laporte, G.: Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks **49**(4), 258–272 (2007)
22. Sexton, T.R., Bodin, L.D.: Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. Transp. Sci. **19**(4), 378–410 (1985)