OR Spectrum (2022) 44:87–119
https://doi.org/10.1007/s00291-021-00656-7

ORIGINAL ARTICLE

An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization

Christian Pfeiffer[1] · **Arne Schulz**[1]

Received: 19 August 2020 / Accepted: 28 September 2021 / Published online: 5 November 2021

Abstract
The paper investigates the static dial-a-ride problem with ride and waiting time minimization. This is a new problem setting of significant practical relevance because several ride-sharing providers launched in recent years in large European 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 present an adaptive large neighbourhood search (ALNS) as well as a dynamic programming algorithm (DP), which are tested in comprehensive computational studies. Although the DP can only be used for a single tour and, due to the computational effort, as a restricted version or for small instances, the ALNS also works efficiently for larger instances. The results indicate that ride-sharing proposals may help to solve the trade-off between individual transport, profitability of the provider, and reduction of traffic and pollution.

Keywords Demand responsive transport · Dial-a-ride · Adaptive large neighbourhood search · Dynamic programming

1 Introduction

In major cities, people enjoy a wide range of transportation modes: buses, urban railways, trams, taxis, and their own private cars. Besides, different ride-sharing providers started in recent years, although the dial-a-ride concept is already known since around 1970 (Oxley 1980). An important difference to the first concepts is that nowadays providers usually use apps to connect drivers and passengers. The biggest company, Uber, which connects private drivers to potential

✉ Arne Schulz
arne.schulz@uni-hamburg.de

Christian Pfeiffer
christian.pfeiffer@uni-hamburg.de

[1] Institute of Operations Management, University of Hamburg, Hamburg, Germany

customers, started 2009 and operates worldwide (Uber Technologies, Inc. 2020). In contrast to Uber, most providers have employed drivers. Among others even companies with competing business models push into the ride-sharing market. MOIA—a subsidiary company of the automobile manufacturer Volkswagen AG—operates in Hanover (since 2018), and Hamburg (since 2019) (MOIA GmbH 2020). ioki—a subsidiary company of the main German railway company, Deutsche Bahn AG—was founded in 2017 and operates among others in Hamburg (ioki GmbH 2020).

Although the dial-a-ride problem is well investigated (Ho et al. 2018), to the best of our knowledge no paper considers the problem these competitors of taxis and private cars in large cities have. For them customer satisfaction is much more important than a minimal driven distance. On the one hand, they compete directly with other modes of transportation and need a high service level to reach a sufficient market share. Thus, they have to be fast enough from the perspective of many customers in order to become a viable alternative for the end user. On the other hand, these companies want to improve the quality of life by less traffic and reduced pollution. Beside the traffic reduction due to the pooling effect, they use electric vehicles to reach this aim. However, electric vehicles are cheaper to recharge and, perspectively [MOIA plans to use autonomous vehicles in Hamburg from 2025 on (MOIA GmbH 2020)], driver costs can be omitted if autonomous vehicles are available. Because of that pure distance minimization is not an appropriate objective. Instead, these ride-sharing providers aim at the minimization of the time between request and arrival at the delivery location for each customer.

In this paper, we minimize the sum of relative detours over all customers in a static setting, i.e. the arrival time at the destination minus the request time minus the direct travel time between pickup and delivery location divided by the latter. The maximum relative detour is penalized additionally to prevent prohibitively long ride or waiting times. We assume a number of vehicles with a maximum passenger capacity as given. Due to our objective, we can omit time windows and maximum waiting and ride time constraints (Psaraftis 1980; compare Cordeau 2006; Ropke et al. 2007). Therefore, we do not need feasibility tests (see Hunsaker and Savelsbergh 2002; Tang et al. 2010; Gschwind and Drexl 2019) since picking up a customer and driving directly to their destination before the next customer gets into the vehicle is always feasible even if only one vehicle with a sufficient capacity is used (all customers must be picked up after their request time). Our objective might lead to empty trips, which are detrimental relating to other objectives such as cost minimization, but they are necessary to compete against taxis and private cars. Taxis have significant empty trips as well (Bischoff and Maciejewski 2016), which are even increased to reduce passenger waiting times (Lees-Miller and Wilson 2012).

We present a literature review in Sect. 2. Afterwards, we introduce the problem in detail in Sect. 3. In Sect. 4, we put forward a general heuristic solution approach based on adaptive large neighbourhood search and in Sect. 5 we introduce a dynamic programming approach for single-tour instances. Both approaches are evaluated in a computational study (Sect. 6) afterwards. The paper closes with a conclusion (Sect. 7).

2 Literature review

The static dial-a-ride problem (DARP) has been subject to an extensive amount of research. Originally, it was formulated by Psaraftis (1980). The exact definition of the DARP differs, but generally the DARP features customer satisfaction (Parragh et al. 2008) to distinguish it from a standard pickup and delivery problem. However, the way customer satisfaction is measured differs a lot. Most commonly used are maximum ride time constraints (Lehuédé et al. 2014) and time windows. Some authors consider hard time windows, which are never allowed to be violated (Cubillos et al. 2014), while others allow violations and penalize them in the objective function (Chevrier et al. 2012; Atahran et al. 2014).

The objective function of a DARP formulation often aims to minimize operating costs like the number of needed vehicles (Baugh et al. 1998) or the total travelled distance (Diana and Dessouky 2004). Mostly, they are combined in a multi-criteria objective function (Baugh et al. 1998). Other approaches search for pareto-optimal frontiers of solutions instead (Atahran et al. 2014).

However, there are also works which consider customer satisfaction in the objective function. Parragh et al. (2009) propose a multi-objective DARP where both the routing cost and the mean user ride times are minimized. The same objective function is evaluated in Molenbruch et al. (2017b). Sexton and Bodin (1985) consider a problem where the customers can specify a desired delivery time and the objective function then minimizes the sum of excess ride times and the deviation to the desired delivery time. Diana and Dessouky (2004) present a DARP where the customer can specify a pickup or a delivery time. The objective function is a weighted sum of the total distance, the excess ride time, and the total idle time. A lexicographic objective is considered by Schilde et al. (2011): Primarily, the total tardiness is minimized, then the number of vehicles and finally the total duration of the routes. To the best of our knowledge, no other work has presented an objective function that minimizes the weighted relative detour as described in Sect. 3. Instead, if the ride or waiting time is considered at all in the objective, it is generally summed up nominally and not relative to the minimum travel time.

Exact solution methods for the DARP have been studied in the form of branch-and-cut algorithms (Cordeau 2006; Ropke et al. 2007) and branch-and-cut-and-price algorithms (Wilson et al. 1999; Baldacci et al. 2011; Gschwind and Irnich 2015). Heuristic solution methods include metaheuristics such as variable neighbourhood search (Parragh et al. 2010; Muelas et al. 2013), Tabu Search (Cordeau and Laporte 2003), simulated annealing (Baugh et al. 1998; Reinhardt et al. 2013), and adaptive large neighbourhood search (Gschwind and Drexl 2019).

A lot of research has been influenced by real-life problems in the transportation of elderly, handicapped or ill passengers (Jorgensen et al. 2007; Heilporn et al. 2011; Detti et al. 2017). Nevertheless, especially for urban areas dial-a-ride problems have also been studied in the context of commercial ride-sharing and demand responsive transport systems (Jaw et al. 1986; Parragh et al. 2015). Muelas et al. (2013) consider a dial-a-ride problem for on-demand transportation in large cities, but they use routing cost minimization as their objective.

3 Problem description

Our problem differs from the standard DARP formulation (Cordeau and Laporte 2007) in the objective function as well as some constraints. As in the standard formulation we consider

- a number of requests with a given demand, i.e. the number of people who want to travel in common,
- k homogeneous vehicles with fixed capacity starting from the same depot,
- a pickup and a delivery location for each request which have to be visited by the same vehicle and in the sequence pickup before delivery, and
- travel times between each pair of locations.

Ride-sharing providers compete in large cities with different modes of transportation like public transport, taxis or private cars. Hence, a customer-oriented objective is necessary to be attractive for customers. On the other side, it is not that important to minimize the total travel distance, as ride-sharing providers typically use electric vehicles, which are cheaper to recharge, to reduce pollution in cities (we do not consider recharging in this paper, as the vehicles are typically recharged during upfront scheduled breaks in practice). In the medium or long term they further target to use autonomous vehicles (no costs for drivers).

Thus, as distinguished from the standard DARP formulation, we do not minimize the travel distance or travel time. Instead, we minimize the customer inconvenience by minimizing the sum of relative detours over all customers. By detour we mean the difference between arrival time at the delivery location and the sum of request time and direct travel time between pickup and delivery location, i.e. the detour includes the waiting time. Afterwards, the detour is divided by the direct travel time between pickup and delivery location to get the relative detour. In order to prevent a single customer from having a prohibitively long detour, the maximum detour over all customers is additionally penalized.

The objective value is given by

$$\sum_{i \in P} q_i \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} + w \cdot \max_{i \in P} \left(\frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} \right), \tag{1}$$

where q_i is the number of passengers in request i, r_i is the request time (i.e. earliest possible pickup time) of request i, $t_{i,i+n}$ is the direct travel time from the pickup to the delivery location of request i, B_{i+n} is the arrival time at the delivery location of request i, and w is the penalty factor for the maximum detour. Hence, the first part of the objective value measures the sum of relative detours, i.e. the sum of waiting time and riding time for each customer relative to the direct travel time, weighted with the number of passengers. The second part of the objective function penalizes the maximum relative detour. A higher value for the penalty factor w increases the importance of minimizing the maximum relative detour compared to minimizing the sum of all weighted relative detours. Note that the number of passengers q_i is not present in the second term, as the maximum relative detour should be measured

independently of the number of passengers in the request. If q_i would be part of the maximum in (1), the maximum could be determined by a request with a smaller relative detour but a larger number of passengers.

The direct travel time between pickup and delivery location can be understood as the minimum required time for the request. It acts as a benchmark since we assume that customers could use their own private car or a taxi to leave at their earliest pickup time and travel straight to their destination. Any delay caused by waiting or by making a detour for another customer will increase the customer's inconvenience and also increase the objective value. Since the objective value is measured relative to the direct travel time, a request with a small direct travel time can lead to a large increase in the objective value for a moderate detour. We believe that this captures actual customer behaviour, as customers will often put their experienced delay in relation to their expected travel time (Nie 2000).

Figure 1 gives an example for the first part of the objective function. It shows the vehicle's tour on top, the customer's waiting and driving times in the middle, and a time axis as well as the request times ($r_1 - r_3$) in the lower part. First, the vehicle drives from the depot to the pickup location of customer 1 (P_1). As the tour between depot and the first location is not relevant for the objective (we assume that each pickup location is reachable directly from depot until the corresponding request time, i.e. vehicle and driver are available early enough), it is not included in the figure. The vehicle reaches the first pickup location at time r_1. From there it drives to the delivery location of customer 1 (D_1). Thus, the first customer has an objective value of zero. They had no waiting time and travelled directly from the pickup to the delivery location. Afterwards, the vehicle drives to P_2. Until it reaches P_2, customer 2 has to wait from r_2 on. After driving to P_3, the vehicle, with customer 2 in it, has to wait because it reaches P_3 before r_3. However, this waiting time counts as journey time for customer 2 since they sit in the vehicle. Then, customers 2 and 3 drive to D_3, where customer 3 leaves the vehicle, before the vehicle finally reaches the destination of customer 2. The tour between the last delivery location and the depot is not relevant for the objective and, therefore, not included in the figure. Like customer 1 customer 3

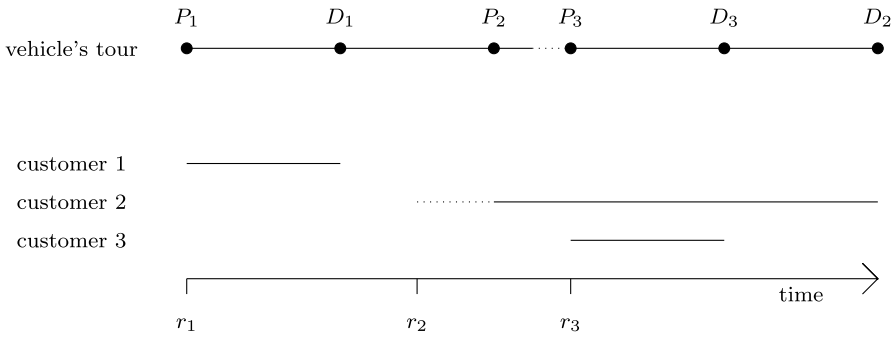

Fig. 1 Example for the first part of the objective function. Continuous lines represent journey times. Dotted lines represent waiting times

has an objective value of zero while customer 2 has a positive objective value. It sums up the waiting time between r_2 and their entrance in the vehicle and the tour $P_2 \to P_3 \to D_3 \to D_2$ (including waiting at P_3). Afterwards, the direct tour $P_2 \to D_2$ is subtracted before the resulting objective value is obtained by dividing the result by the direct travel time $P_2 \to D_2$.

Although we do not minimize the travel time directly, our objective minimizes it indirectly if customers are waiting to be fetched or sitting in the vehicle. Solution approaches will try to be as fast as possible at a customer location to reduce the waiting time of later customers. Thus, the algorithm tries to bundle rides, i.e. increase the pooling rate, to reduce travel distance.

Our objective leads further to adjustments of the constraints in comparison with the standard DARP formulation. Because we minimize customer inconvenience, we do not need constraints which ensure a certain degree of customer satisfaction. Therefore, we abstain from a maximum ride time restriction and time windows. This is also the reason why there is a feasible solution in our problem setting. If the vehicle's capacity is at least as high as the highest customer demand, all solutions which sequence the delivery location of a customer directly after their pickup location are feasible. Although such a solution is minimal according to the journey time of each customer, it might be bad according to the waiting times. Therefore, the proposed objective solves the trade-off between minimal journey time and waiting time by combining the rides of earlier customers, and, thus, increase their journey time, to reduce the waiting time of later customers. The maximum relative ride time is further controlled by the second part of the objective function. By this, we ensure that a single customer is not disadvantaged excessively in favour of the other customers.

In the following, we present a mixed-integer programming formulation for our problem setting. The notation used in the remainder of the paper is:

sets and indices:	
$P = \{1, \dots, n\}$	pickup locations
$D = \{n+1, \dots, 2n\}$	delivery locations
$i, j \in I$	locations, $I = \{0, 2n+1\} \cup P \cup D$
parameters:	
k	number of tours
n	number of requests
Q	capacity of each vehicle
q_i	demand at location $i \in P \cup D$: $q_i > 0$ and $q_{i+n} = -q_i \quad \forall i \in P$
r_i	request time (= earliest possible pickup time) of request $i \in P$ with $r_i = r_{i-n} + t_{i-n,i} \quad \forall i \in D$
t_{ij}	travel time between locations i and $j \in I$
w	Penalty factor for the maximum relative detour of all customers
M_{ij}	sufficiently large value
variables:	
B_i	departure time of location $i \in P \cup D$
Q_i	number of passengers in the vehicle after leaving location $i \in P \cup D$
X_{ij}	= 1 if j is direct successor of i, 0 otherwise; $i, j \in I$

We always consider a travel speed of one distance unit per time unit and, thus, use distances and travel times synonymously. We assume that we have a fleet of k vehicles with passenger capacity Q that each drive a single tour starting at a common depot ($i = 0$). Each tour ends at the end depot ($i = 2n + 1$). Since the return to the end depot does not influence the objective value, the formulation can also be used to model situations where the tours are open-ended. There are n customer requests that all need to be served. Each request i has an earliest pickup time r_i which acts as a hard lower bound for the pickup time. There is no upper limit to the pickup time of a customer, however, large deviations are unlikely since they increase the objective value and are additionally punished by the penalty factor w. All travel times are assumed to fulfil the triangle inequality.

Beside objective function (1), which is minimized, the model is the standard 2-index dial-a-ride formulation by Ropke et al. (2007) without time windows and maximum ride time constraints. We need to define the sets $S \subseteq I$ and $\mathcal{S}$. $\mathcal{S}$ is the set of all sets S for which $0 \in S$ and there is at least one request i that has its delivery node in S but not the pickup node, i.e. $\mathcal{S} = \{S : 0 \in S \wedge \exists i : (i \notin S \wedge n + i \in S)\}$. Then, the constraints are:

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

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

$$\sum_{j \in P \cup D} X_{0j} = k \tag{4}$$

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

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

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

$$B_i \geq r_i \quad \forall i \in P \cup D \tag{8}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \quad \forall i \in P \cup D \tag{9}$$

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

Constraints (2) and (3) enforce that every pickup and every delivery location is entered and left exactly once. Constraint (4) forces the model to leave the depot k times to form tours. Constraints (5) serve both as a subtour elimination constraint

as well as to ensure that pickup and delivery location of each customer are always visited in the same tour and in the correct order. Constraints (6) and (7) enforce the correct setting of time and capacity variables. Finally, the Constraints (8), (9), and (10) restrict the variable domains.

In order to generate the sets S for Constraints (5), we adopt the procedure by Ropke et al. (2007): At every integer feasible solution, a maximum flow problem is run for each customer to verify whether the customer's pickup node and delivery node are visited in the same tour and correct order. If not, a new set S is added to cut off the current solution.

4 Adaptive large neighbourhood search

Adaptive large neighbourhood search (ALNS) is a metaheuristic developed by Ropke and Pisinger (2006a). In each iteration, a new solution is created based on the current solution: first, a removal heuristic removes certain customers, i.e. their pickup as well as their delivery locations, from the solution. Then, an insertion heuristic reinserts the removed customers back into the solution. Both the removal and the insertion heuristic are chosen based on their past performance in the algorithm. If a specific heuristic can consistently create new, high-quality solutions, it is used more often. The new solution is accepted based on the simulated annealing acceptance criterion. In the following, we describe our implementation in detail.

Our algorithm's initial solution is randomly generated. Unless otherwise mentioned, the removal heuristics will always remove q (determined similarly to Ropke and Pisinger (2006a)) customers and the insertion heuristics will insert all of the removed customers back into the solution. The parameter q is dependent on the number of customers n and is drawn from the interval $[1, \xi \cdot n]$ with $0 < \xi < 1$ in each iteration.

The heuristics marked with an * are run with an additional random component that perturbs the selection process so that we do not always choose the q "best" customers but only give a higher probability to better customers to be chosen. It works by drawing a random number $x \in [0, 1)$ and then raising it to the power of a parameter $\rho > 1$. x^ρ is a random number that is weighted towards 0 and if multiplied with the number of options and rounded down, will give the index of the option that should be chosen. If the options are sorted according to a quality criterion beforehand, higher quality options will be chosen more often. The procedure is explained in more detail in Ropke and Pisinger (2006a) and was originally proposed by Shaw (1997). The higher the controlling parameter ρ, the higher the influence of the underlying ordering and the lower the randomization.

The heuristics marked with an $^+$ are also run with an additional random component but instead of the aforementioned rank based selection by Shaw (1997) we run a roulette wheel selection where the quality criteria of all the options are summed up and each criterion is then divided by the sum to generate the probability of the option to be selected.

4.1 Removal neighbourhoods

We implemented the following removal neighbourhoods:

1. *Random removal*: q customers are selected randomly and removed from their tour [diversification; Pisinger and Ropke 2007].
2. *Worst removal (tour)*$^+$: the q customers with the worst contribution to their tour are removed, i.e. the q customers for which the difference between the objective value of the corresponding tour with and without the customer is maximized (Pisinger and Ropke 2007).
3. *Worst removal (individual)*$^+$: the q customers with the worst contribution to the objective value are removed, i.e. the q customers with the highest individual objective value. The request with the current maximum relative detour is additionally penalized by adding the product of w and its contribution as well.
4. *Related removal distance**: choose a random customer $i \in P$ and remove it as well as their $q-1$ most related customers, i.e. those customers for which

$$s_{ij} = t_{ij} + t_{ji} + t_{i,j+n} + t_{j+n,i} + t_{j,i+n} + t_{i+n,j} + t_{i+n,j+n} + t_{j+n,i+n} \tag{11}$$

   with $j \in P$ is minimal (Shaw 1998; Pisinger and Ropke 2007).
5. *Related removal request time**: referred to as *time-oriented removal* in (Pisinger and Ropke 2007), this heuristic works the same as in 4. with

$$s_{ij} = |r_i - r_j|.$$

6. *Proximity removal*$^+$: Compute

$$s_{ij} = \begin{cases} \frac{1}{t_{ij}+t_{ji}+t_{i,j+n}+t_{j+n,i}+t_{j,i+n}+t_{i+n,j}+t_{i+n,j+n}+t_{j+n,i+n}+|r_i-r_j|} & \text{if } q_i + q_j \leq Q, \\ 0 & \text{else} \end{cases}$$

   and take for each customer $i \in P$ the highest s_{ij} value over all customers $j \in P$ which are assigned to different tours. Then, choose q customers according to a roulette-wheel selection where the weight of customer i is the highest s_{ij} value. In contrast to related removal, the removed customers are not similar amongst each other but have a similar request in a different route in which they might be inserted (Molenbruch et al. 2017a). As *Proximity removal* does not work in instances with only a single tour (i.e. $k = 1$), it is automatically turned off in such cases.
7. *Cluster removal distance*: select a random tour. Apply Kruskal's algorithm (Kruskal 1956) on all customers of the tour until two clusters are left. The edge costs for the algorithm are given by

$$\bar{t}_{ij} = t_{ij} + t_{i+n,j+n}$$

   for all i, j in the considered tour. Choose one of the two clusters randomly and remove all its customers (Ropke and Pisinger 2006b).

4.2 Insertion neighbourhoods

1. *Basic greedy insertion*: all customers are sorted according to the costs of their best insertion. The customer with the best insertion (smallest increase in the objective value) is inserted first. Afterwards, the next best customer is chosen. This is repeated until all customers are inserted (Pisinger and Ropke 2007).
2. *Greedy roulette insertion*$^+$: each feasible insertion for every customer is evaluated. Then, a roulette wheel selection is run where each insertion is weighted with the inverse of its objective value, i.e. insertions with lower costs have a higher chance to be chosen and the selection probability depends on the quality of all other options. This increases diversification compared to the normal greedy insertion.
3. *Randomized greedy insertion**: all customers are sorted according to the costs of their best insertion. In contrast to insertion neighbourhood 1., we do not simply choose the best customers first but run a weighted random selection with the randomization from Shaw (1997) instead (described above in the second to last paragraph before Sect. 4.1). Compared to insertion neighbourhood 2., the randomized greedy insertion only takes the ordering, i.e. the ranking, of the insertions into account, not the actual differences between their costs.
4. *Regret g insertion*: the customer with the highest regret value is inserted, that is, the customer that has the biggest difference between the cost of the insertion into the best tour and the insertions into the g best tours (Pisinger and Ropke 2007). If $g > 2$, the differences between the best and the other tours are added up to compute the regret value. In instances with fewer tours than the specified regret level, the level is automatically adjusted to the number of tours.
5. *Greedy insertion perturbation*: the same as insertion neighbourhood 1., but each objective value is multiplied with a random factor in $[1-\gamma, 1+\gamma]$ with $0 < \gamma < 1$. Afterwards, the best solution is chosen (Hemmelmayr et al. 2012). This operator allows us to observe the effect of noise perturbation for a single heuristic.

4.3 Choice of removal and insertion heuristic

We follow the original implementation of Ropke and Pisinger (2006a) where heuristics are chosen by roulette wheel selection and each heuristic is given a score whenever a new solution is found: σ_1 is assigned to a heuristic whenever a new best solution is found, σ_2 when a better solution is found that has not been accepted before and σ_3 when a worse but accepted solution is found which has not been accepted before. After a segment of 100 iterations, the weights for the roulette wheel selection are updated with the scores and the scores are reset. A reaction factor $\phi \in [0, 1]$ determines how quickly the heuristics change their weights. A higher ϕ will increase the influence of the performance in the current segment and decrease the influence of the previous weight. The determination of parameters is explained in Sect. 6.2.

4.4 Temperature

Our algorithm is run for 25,000 iterations or until an objective value of 0 is found. At the end of each iteration, we set the temperature T to $T := \alpha T$ with $0 < \alpha < 1$. The starting temperature is generated similarly to Ropke and Pisinger (2006a): We use the objective value of the initial solution and set the starting temperature such that a hypothetical move that worsens the initial solution by a factor of ω (with $\omega \geq 0$) is accepted with a probability of 50%. As an example, for $\omega = 0.2$ a solution that is 20% worse than the initial solution is accepted with a probability of 50%. The temperature is used in the simulated annealing acceptance criterion (Kirkpatrick et al. 1983): Whenever a new solution is found that is better than the old one, it is accepted. If it is worse, it is accepted with a probability given by $e^{-\Delta/T}$ where Δ is the difference between the new and the old objective value and T is the current temperature.

5 Dynamic programming algorithm

Dynamic programming algorithms have been studied for dial-a-ride problems for decades. Early approaches were given by Psaraftis (1980, 1983). Our approach is based on the dynamic programming algorithm for the travelling salesman problem by Bellman (1962) and Held and Karp (1962). In a recent paper, Ritzinger et al. (2016) developed a dynamic programming algorithm for the standard dial-a-ride formulation with time windows and maximum ride time constraints. Due to these further constraints, they need a larger state space than we do. The following algorithm finds an optimal solution for any single tour instance (i.e. $k = 1$) in our problem setting. Although the DP can also be expanded to multiple tours, we refrain from doing so because of the memory requirements and computation times the single-tour DP already needs (compare Table 4).

- *Stages* number of locations without the depot $I \backslash \{0\}$.
- *States* states are described by the quadruple (S, v, t, d_{max}), where v is the current last location, S is the set of all already scheduled locations without v, t the point of time, and d_{max} the maximum relative detour over all customers so far. The number of customers in the vehicle is not part of the state, as it can be calculated with v and S as $\sum_{i \in S \cup \{v\}} q_i$. The initial state is $(\{\}, 0, 0, 0)$, i.e. no customer has been visited so far ($S = \{\}$), the current location is the depot (0), the current point in time is 0 ($t = 0$), and the maximum relative detour is 0.
- *Transitions* the tour is expanded by one location v' from the set $\{i \in P : i \notin S \cup \{v\} \wedge q_i + \sum_{j \in S \cup \{v\}} q_j \leq Q\} \cup \{i \in D : i \notin S \cup \{v\} \wedge i - n \in S \cup \{v\}\}$, i.e. all pickup locations are allowed to be scheduled which have not been scheduled so far as long as q_i does not exceed the remaining capacity $Q - \sum_{j \in S \cup \{v\}} q_j$ as well as all delivery locations for which the corresponding pickup location has already

been scheduled and which have not been scheduled themselves. The time t increases by the travelled time $t_{vv'}$ between locations v and v' or up to $r_{v'}$ if $v' \in P$ and $t + t_{vv'} < r_{v'}$, i.e. $t' = \max(t + t_{vv'}, \mathbb{1}_{\{v' \in P, t+t_{vv'} < r_{v'}\}} \cdot r_{v'})$, where $\mathbb{1}_{\{v' \in P, t+t_{vv'} < r_{v'}\}}$ is 1 if $v' \in P$ and $t + t_{vv'} < r_{v'}$ and 0 else. Thereby, $d'_{max} = \max\left(d_{max}, \max_{i \in P: i+n \notin S} \frac{t' - r_i - t_{i,i+n}}{t_{i,i+n}}\right)$. While the first part of the maximum covers the case that the maximum relative detour does not change, the second part covers the case where a not yet delivered customer leads to a new maximal relative detour. Hence, the new state is $(S \cup \{v\}, v', t', d'_{max})$.

- *Recursion* costs of a state (S, v, t, d_{max}) are

$$\begin{aligned} C(S, v, t, d_{max}) &= \min_{\bar{v} \in S} \Big\{ C(S \backslash \{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max}) \\ &\quad + \sum_{i \in P: i+n \notin S} \mathbb{1}_{\{t > r_i + t_{i,i+n}\}} \cdot q_i \cdot \frac{\min(t - \bar{t}, t - r_i - t_{i,i+n})}{t_{i,i+n}} \\ &\quad + w \cdot \max(0, d_{max} - \bar{d}_{max}) \Big\} \end{aligned} \tag{12}$$

with $\mathbb{1}_{\{t > r_i + t_{i,i+n}\}} = \begin{cases} 1 \text{ if } t > r_i + t_{i,i+n}, \\ 0 \text{ else} \end{cases}$ and $t = \begin{cases} \bar{t} + t_{\bar{v}v} & \text{if } v \in D, \\ \max\{\bar{t} + t_{\bar{v}v}, r_v\} & \text{if } v \in P. \end{cases}$

The time in the state description leads to a potentially high number of states. Nevertheless, it is necessary, as the tour $\tilde{v} \to i + n \to j \to v$ might be longer (let t be the time the vehicle reaches location v) than $\tilde{v} \to j \to i + n \to v$ but leads to a smaller objective value, as customer i is taken away earlier. In contrast, the longer tour increases the relative detour of all other customers $j \in P$ with $r_j < t$ which have not been taken away so far. Hence, it is possible that $C(S, v, t, d_{max}) < C(S, v, t', d'_{max})$ although $t > t'$. However, we get a dominance rule for the case $t' \geq t$.

Theorem 1 *Let*

$$\begin{aligned} C(S, v, t, d_{max}) &+ \sum_{i \in P: i+n \notin S \cup \{v\}} q_i \cdot \frac{\max(t' - \max(t, r_i + t_{i,i+n}), 0)}{t_{i,i+n}} \\ &+ w \cdot \max(0, d'_{max} - d_{max}) \\ &< C(S, v, t', d'_{max}). \end{aligned}$$

Then, (S, v, t, d_{max}) *dominates* (S, v, t', d'_{max}) *if* $t \leq t'$.

Proof Both states (S, v, t, d_{max}) and (S, v, t', d'_{max}) coincide in the already visited customer locations. Let $\Pi = \pi_{|S|+2}, \ldots \pi_{2n}$ be any sequence of the not yet visited customers. Let $t \leq t'$. Let B_i (B'_i) be the departure time of location $i \in I$, if we merge state (S, v, t, d_{max}) ((S, v, t', d'_{max})) and Π to a complete solution and wait during the tour only if necessary. Then, $B_i \leq B'_i$ for all customer locations in Π. Costs for customers with $i + n \in S \cup \{v\}$ do not change anymore. Costs can occur for customers with $i + n \notin S \cup \{v\}$ if the vehicle has to wait longer if starting at t than if starting at t'. These costs are included in $\sum_{i \in P: i+n \notin S \cup \{v\}} q_i \cdot \frac{\max(t' - \max(t, r_i + t_{i,i+n}), 0)}{t_{i,i+n}}$. Costs for driving

the partial tour Π occur for both states. Thus, the relative detour costs starting in (S, v, t, d_{max}) are at most the relative detour costs starting in (S, v, t', d'_{max}) plus $\sum_{i \in P: i+n \notin S \cup \{v\}} q_i \cdot \frac{\max(t'-\max(t, r_i+t_{i,i+n}), 0)}{t_{i,i+n}}$.

Because of $C(S, v, t, d_{max}) + w \cdot \max(0, d'_{max} - d_{max}) < C(S, v, t', d'_{\max})$, it holds

$$C(S, v, t, \max(d_{\max}, d'_{\max})) < C(S, v, t', d'_{\max}).$$

Since $B_i \leq B'_i$ for all customer locations in Π, costs for the maximum relative detour do not increase more strongly at any point of the tour if we start in state $(S, v, t, \max(d_{\max}, d'_{\max}))$ than in state $(S, v, t', d'_{\max})$. In total, completing the tour from v at t leads to no higher costs than starting in v at t'.

Theorem 2 views a variant of Theorem 1 where $t' \geq t \geq \max_{i \in P}(r_i + t_{i,i+n})$.

Theorem 2 *Let* $C(S, v, t, d_{\max}) + w \cdot \max(0, d'_{max} - d_{max}) < C(S, v, t', d'_{max})$. *Then,* (S, v, t, d_{max}) *dominates* (S, v, t', d'_{max}) *if* $t' \geq t \geq \max_{i \in P}(r_i + t_{i,i+n})$.

Proof Let Π again be any feasible sequence of the remaining locations $i \in (P \cup D) \backslash (S \cup \{v\})$. $t' \geq t \geq \max_{i \in P}(r_i + t_{i,i+n})$ implies that $\frac{t - r_i - t_{i,i+n}}{t_{i,i+n}} > 0$ for all $i \in P : i + n \notin S \cup \{v\}$ immediately when the car leaves v. Thus, the car does not wait during the partial tour Π and $B'_i = B_i + (t' - t)$ for all customer locations in Π. Hence, completing the tour from v at t leads to no higher costs than starting in v at t' with the same argumentation as in the proof of Theorem 1 if $t \leq t'$ and

$$C(S, v, t, d_{max}) + w \cdot \max(0, d'_{max} - d_{max}) < C(S, v, t', d'_{max}).$$

Theorem 3 considers also the case, where all customers have a positive relative detour immediately when the car moves $(\min(t, t') \geq \max_{i \in P}(r_i + t_{i,i+n}))$. Thus, it generalizes Theorem 2.

Theorem 3 *Let*

$$\begin{aligned} C(S, v, t, d_{max}) + w \cdot \Big(&\max(0, d'_{max} - d_{max}) \\ &+ \max\left(0, \max_{i \in P: i+n \notin S \cup \{v\}} \frac{t - t'}{t_{i,i+n}}\right)\Big) \\ &< C(S, v, t', d'_{max}). \end{aligned}$$

Then, (S, v, t, d_{max}) *dominates* (S, v, t', d'_{max}) *if* $\min(t, t') \geq \max_{i \in P}(r_i + t_{i,i+n})$.

Proof We assume $t > t'$. Otherwise, we have the situation of Theorem 2. Let Π again be any feasible sequence of the remaining locations $i \in (P \cup D) \backslash (S \cup \{v\})$. Then, $B_i = B'_i + (t - t')$ for all of them. $t > t' \geq \max_{i \in P}(r_i + t_{i,i+n})$ implies that $\frac{t' - r_i - t_{i,i+n}}{t_{i,i+n}} > 0$ for all $i \in P : i + n \notin S \cup \{v\}$ immediately when the car leaves v. However,

$t > t' \geq \max_{i \in P}(r_i + t_{i,i+n})$ implies also that $\frac{t - r_i - t_{i,i+n}}{t_{i,i+n}} = \frac{t' - r_i - t_{i,i+n}}{t_{i,i+n}} + \frac{t - t'}{t_{i,i+n}}$. Hence, the costs for the longer relative detours in (S, v, t, d_{max}) in comparison to (S, v, t', d'_{max}) are already included in $C(S, v, t, d_{max})$. Thus, $C(S, v, t, d_{max}) + \sum_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - t}{t_{i,i+n}} < C(S, v, t', d'_{max}) + \sum_{i \in P: i+n \notin S \cup \{v\}} \frac{B'_{i+n} - t'}{t_{i,i+n}}$.

The so far maximum relative detours d_{max} and d'_{max} are also priced in in $C(S, v, t, d_{max})$ and $C(S, v, t', d'_{max})$, respectively. Because of $d_{max} + \max(0, d'_{max} - d_{max}) = \max(d_{max}, d'_{max})$, we get

$$\begin{aligned} C(S, v, t, d_{max}) + w \cdot \max(0, d'_{max} - d_{max}) &= C(S, v, t, \max(d_{max}, d'_{max})) \\ &< C(S, v, t', d'_{max}). \end{aligned}$$

Since $B_i = B'_i + (t - t')$ for all $i \notin S \cup \{v\}$,

$$\max_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} > \max_{i \in P: i+n \notin S \cup \{v\}} \frac{B'_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}}.$$

If $\max(d_{max}, d'_{max}) \geq \max_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}}$, we are finished, as the larger maximal relative detour is already priced in at location v ($C(S, v, t, \max(d_{max}, d'_{max})) < C(S, v, t', d'_{max})$). So, let $\max_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} > \max(d_{max}, d'_{max})$. We distinguish two cases:

1. $\max_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} > \max(d_{max}, d'_{max}) \geq \max_{i \in P: i+n \notin S \cup \{v\}} \frac{B'_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}}$
2. $\max_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} > \max_{i \in P: i+n \notin S \cup \{v\}} \frac{B'_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} > \max(d_{max}, d'_{max})$.

It holds

$$\begin{aligned} &\max_{i \in P: i+n \notin S \cup \{v\}} \frac{B_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} \\ &= \max_{i \in P: i+n \notin S \cup \{v\}} \left(\frac{B'_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} + \frac{t - t'}{t_{i,i+n}} \right) \\ &\leq \max_{i \in P: i+n \notin S \cup \{v\}} \frac{B'_{i+n} - r_i - t_{i,i+n}}{t_{i,i+n}} + \max_{i \in P: i+n \notin S \cup \{v\}} \frac{t - t'}{t_{i,i+n}}. \end{aligned}$$

In the first case, this leads immediately to

$$
\begin{aligned}
&C\left(S, v, t, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{B_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}\right)\\
&\quad= C\left(S, v, t, \max_{i\in P: i+n\notin S\cup\{v\}} \left(\frac{B'_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}+\frac{t-t'}{t_{i,i+n}}\right)\right)\\
&\quad\le C\left(S, v, t, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{B'_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}\right)\\
&\qquad + w\cdot\max\left(0, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{t-t'}{t_{i,i+n}}\right)\\
&\quad\le C\big(S, v, t, \max(d_{max}, d'_{max})\big) + w\cdot\max\left(0, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{t-t'}{t_{i,i+n}}\right)\\
&\quad= C(S, v, t, d_{max}) + w\cdot\left(\max(0, d'_{max}-d_{max}) + \max\left(0, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{t-t'}{t_{i,i+n}}\right)\right) \qquad (13)
\end{aligned}
$$

$$
< C(S, v, t', d'_{max}). \qquad (14)
$$

In the second case,

$$
\begin{aligned}
&C\big(S, v, t, d_{max}\big) + w\cdot\max\left(0, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{t-t'}{t_{i,i+n}}\right)\\
&\quad\le C\big(S, v, t, \max(d_{max}, d'_{max})\big) + w\cdot\max\left(0, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{t-t'}{t_{i,i+n}}\right)\\
&\quad\overset{(13),(14)}{<} C(S, v, t', d'_{max}) \qquad (15)
\end{aligned}
$$

implies

$$
\begin{aligned}
&C\left(S, v, t, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{B_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}\right)\\
&\quad= C\left(S, v, t, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{B'_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}+\frac{t-t'}{t_{i,i+n}}\right)\\
&\quad\le C\left(S, v, t, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{B'_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}\right)\\
&\qquad + w\cdot\max\left(0, \max_{i\in P: i+n\notin S\cup\{v\}} \frac{t-t'}{t_{i,i+n}}\right)\\
&\quad\overset{(15)}{<} C\left(S, v, t', \max_{i\in P: i+n\notin S\cup\{v\}} \frac{B'_{i+n}-r_i-t_{i,i+n}}{t_{i,i+n}}\right).
\end{aligned}
$$

Thus, we can add the costs for the relative detour of each customer $i \in P : i+n \notin S\cup\{v\}$ as well as the maximal detour costs for any sequence Π

of the remaining customers to $C(S, v, t, d_{max})$ (and $C(S, v, t', d'_{max})$) such that state (S, v, t, d_{max}) leads to no higher cost at the end of the tour than state (S, v, t', d'_{max}).

Theorem 4 dominates states at pickup locations v where the car has to wait if it is possible to use the waiting time to bring a customer who is in the car to the delivery location or pick up another customer and drive to their delivery location while still reaching v at time r_v.

Theorem 4 *Let* $(S\backslash\{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max})$ *be the predecessor of* (S, v, t, d_{max}) *with* $v \in P$ *and* $t = r_v$, *i.e.* $(S\backslash\{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max})$ *determines the costs of* (S, v, t, d_{max}) *in* (12). *Then,* (S, v, t, d_{max}) *is dominated if one of the sets* $\{i \in P \cap S : i + n \notin S \wedge \bar{t} + t_{\bar{v},i+n} + t_{i+n,v} \leq t\}$ *and* $\{i \in P\backslash(S \cap P) : \max(r_i, \bar{t} + t_{\bar{v},i}) + t_{i,i+n} + t_{i+n,v} \leq t \wedge \sum_{j \in S} q_j + q_i \leq Q\}$ *is not empty, i.e. if it is possible to take away a customer in the vehicle or pick up and take away another customer and still reach location v not later than* r_v.

Proof First, we compare the two transitions $(S\backslash\{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max}) \rightarrow (S, v, t, d_{max})$ and $(S\backslash\{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max}) \rightarrow (S, i + n, t', d'_{max}) \rightarrow (S \cup \{i\}, v, t, \tilde{d}_{max})$ with $i \in \{i \in P \cap S : i + n \notin S \wedge \bar{t} + t_{\bar{v},i+n} + t_{i+n,v} \leq t\}$. In both transitions $q_j \cdot \frac{\min(t - r_j - t_{j,j+n})}{t_{j,j+n}}$ is the same for all $i \neq j \in P$. But customer i reaches the delivery location at no later point in time in the second transition, as $t' \leq t$, i.e. $q_i \cdot \frac{\min(t' - r_i - t_{i,i+n})}{t_{i,i+n}} \leq q_i \cdot \frac{\min(t - r_i - t_{i,i+n})}{t_{i,i+n}}$. As $q_j \cdot \frac{\min(t - r_j - t_{j,j+n})}{t_{j,j+n}}$ is for no customer $j \in P$ larger in the second transition than in the first, $\tilde{d}_{max} \leq d_{max}$. So, the first transition is dominated.

By the same argumentation $(S\backslash\{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max}) \rightarrow (S, i, t^1, d^1_{max}) \rightarrow (S \cup \{i\}, i + n, t^2, d^2_{max}) \rightarrow (S \cup \{i, i + n\}, v, t, \hat{d}_{max})$ dominates $(S\backslash\{\bar{v}\}, \bar{v}, \bar{t}, \bar{d}_{max}) \rightarrow (S, v, t, d_{max})$ for a customer $i \in \{i \in P\backslash(S \cap P) : \max(r_i, \bar{t} + t_{\bar{v},i}) + t_{i,i+n} + t_{i+n,v} \leq t \wedge \sum_{j \in S} q_j + q_i \leq Q\}$.

Especially the last dominance criterion (Theorem 4) leads to a reduction of states in early stages because customers with high request times are dominated.

The above formulation can solve smaller instances but for larger numbers of customers the time and memory requirements make this exact procedure unsuitable. Therefore, we change the DP to a restricted DP similar to Malandraki and Dial (1996) and Ritzinger et al. (2016). At each stage we only branch on the B states with the lowest costs. If B is sufficiently large, this restricted DP still maintains the guarantee of optimality but if there are more than B states in a stage, it devolves into a heuristic.

6 Computational study

The algorithms were implemented in C++ and compiled with Visual Studio 2017. The tests were run on an AMD Ryzen Threadripper 3990X with 2.9GHz (single-threaded) and 256GB of RAM. The mixed integer program (MIP) was solved with CPLEX version 12.9 with dynamic search enabled.

6.1 Instances

The instances in the computational study are synthetic test instances which are based on a practical application in a district of Hamburg, Germany. For each instance, the pickup and delivery locations are determined randomly out of a set of possible stops of a ride-sharing provider. Since the city layout needs to be taken into account, the resulting distance matrix is not symmetrical but fulfils the triangle inequality. Distances are rounded to the second decimal place and are measured in units of time. The largest possible travel time between two locations is roughly 28 minutes. The smallest possible request time is determined by the maximum distance of the depot to any pickup location $maxT$, i.e. $maxT = \max_{i \in P} t_{0,i}$. This ensures that every request can be reached on time if it is the first location in a tour. The request times are uniformly distributed in the interval $[maxT, maxT + \beta \cdot n]$ with $\beta \geq 0$. For our instances we set $\beta = 100$ which resulted in instances of adequate difficulty as preliminary tests showed. Larger values for β tend to spread out the requests too much so that the instances become too easy while smaller values lead to an unrealistically high concentration of requests in a tight time window. The vehicle capacity is set to six passengers which is common in practice.

The number of tours is varied between 1, 2, 5, and 8. The number of requests is set such that the ratio of requests to tours (i.e. n/k) equals 5, 10, 15, and 20 for each tour configuration. This leads to 16 parameter configurations ranging from 5 to 160 requests. For each configuration we generated 100 instances. Half of the instances had their q_i set to 1 for every customer. In the other half we drew each q_i according to an exponential distribution with $\lambda = 0.9$ and rounded the result to the next highest integer. We redrew if the result was greater than our passenger capacity of six. The 400 instances with one tour were also used for the evaluation of the DP. All utilized instances can be found under the following link: https://doi.org/10.25592/uhhfdm.9610.

6.2 Parametrization

The instance generator described above was also used for the parametrization but we took care not to reuse any instances. Furthermore, the instances for the parametrization had passenger demands of one per request and no single-tour instances as the latter makes some of the heuristics unsuitable. Additionally, we varied the penalty factor w in the used instances between 0, n, $2n$, and $5n$ such that the tuning is not biased towards one specific value of w.

The parameter values for ALNS were determined with the help of the *irace* package which can automatize the parameter tuning (López-Ibáñez et al. 2016). *irace* is given a set of parameters and their respective ranges as well as a training set of instances. Then, it performs a certain amount of experiments where the algorithm is tried out with different parameter configurations. The distributions that *irace* uses for determining the parameters are changed throughout the search

to focus on the more promising areas in the parameter search space. In the end, *irace* gives out a set of elite configurations that have performed best. We chose the best of these elite configurations which resulted in the use of the parameter values in Table 1. The third column additionally gives the range for the distributions that *irace* used.

In a next step, we are interested in finding a suitable value for the penalty factor w. Preliminary results showed that a value of $w = n$ results in a reasonable trade-off between the reduction of the maximum relative detour and the reduction of the sum of all weighted relative detours. More details about this preliminary study can be found in Appendix A. As a result, all instances in the following use the penalty factor $w = n$.

While the adaptive procedure in the ALNS is in theory capable of identifying poorly performing heuristics during the search, in practice a preselection of heuristics can improve performance significantly (François et al. 2016). For this reason, we performed a series of tests to identify a good heuristic configuration. A more detailed description of the tests for the heuristic configurations can be found in Appendix B.

The evaluation of the heuristic configurations has shown that there are two competitive configurations which are displayed in Table 2. Both of these configurations were utilized in the following calculations. The instances we used for the parameter tuning are different from the ones in the computational study to avoid any kind of bias.

6.3 Discussion of results

For the ALNS we solved each of the 1,600 instances of our test bed with the two configurations of Table 2. For the 400 instances with one vehicle ($k = 1$) we turned off the *proximity removal* heuristic from the first ALNS configuration, as it does not work if only a single tour is examined. Furthermore, these 400 instances were solved by the restricted DP where we varied the amount of states. As an additional benchmark, we also ran the MIP for the smaller instances with a computational time of

Table 1 Parameter values for ALNS

Parameter	Value	Interval for the parameter distribution
α	0.9989	[0.995, 0.999999]
ω	1.5165	[0.01, 2]
σ_1	32	[10, 50]
σ_2	14	$[0, \sigma_1]$
σ_3	2	$[0, \sigma_2]$
ξ	0.2162	[0.05, 0.5]
ϕ	0.4964	[0.01, 0.99]
γ	0.3544	[0.1, 1.9]
ρ	6.3698	[1.01, 10]

Table 2 Used heuristics of the two best configurations found in the preliminary tests

	1	2
Random removal	•	•
Worst removal (tour)	•	•
Worst removal (individual)	•	•
Related removal distance		•
Related removal request time	•	
Proximity removal	•	
Cluster removal		
Basic greedy insertion	•	
Greedy roulette insertion	•	•
Randomized greedy insertion	•	•
Regret 2 insertion		
Regret 3 insertion	•	•
Greedy insertion perturbation	•	•

one hour. In any of the following tables, z refers to the (average) objective value and *Time* refers to the (average) computational time in seconds.

6.3.1 Results of ALNS

In Fig. 2, the average relative difference to the best found configuration for both configurations is displayed. This means that we computed the relative difference of the worse to the better configuration for all 100 tested instances and averaged them afterwards. The results for the single tour instances are not shown because of the aforementioned changes to the configuration and the separate evaluation in the next subsection. In general, both configurations are fairly close to each other with no clear apparent winner. Comparing the mean over all instances (again excluding the instances with $k = 1$), configuration 2 is on average 0.23% better. Since the results between the two configuration are so close, we only report the values for the second configuration in the following.

A summary of the results of the second configuration can be found in Table 3. The table also states the results of the MIP for comparison. We did not attempt to solve the larger instances with the MIP for $k = \{2, 5, 8\}$ as the largest ones we did run could only be solved to feasibility in rare cases (compare $(k, n) = (2, 20), (5, 25)$ and $(8, 40)$). Computation times for ALNS increase with a higher amount of customers, specifically if there are few vehicles, as longer tours prolong the search for a good insertion position in an iteration. Nevertheless, the computation times for ALNS are still acceptable in a static setting. The mixed integer model performs very poorly and can only reliably solve very small instances ($n = 5$) to optimality in the time limit of one hour. Additionally, the lower bounds for slightly larger instances are very close to 0 which is likely due to a weak LP relaxation of the model formulation. In summary, the basic MIP formulation does

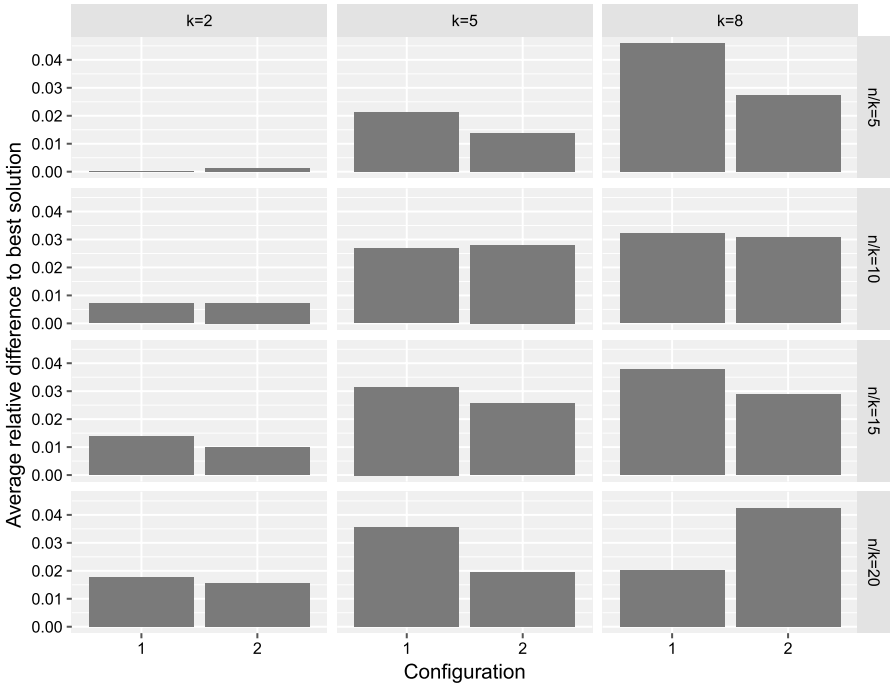

Fig. 2 A comparison between the two different heuristic configurations for ALNS. Each square shows the average relative deviation from the best found solution of 100 instances

not seem suitable to solve the problem which further emphasizes the need for a heuristic like ALNS.

The first part of the objective value for the ALNS results is further visualized in a histogram in Fig. 3 where the first part of the objective value is divided by the number of passengers. The resulting value is the average relative detour per passenger. For the instances where we set $q_i = 1$ for all requests, the first part of the objective value is simply divided by n, but for the other half of the instances, the total number of passengers varies due to the random generation. It is apparent in Fig. 3 that the detour per passenger is especially high for instances with only one vehicle ($k = 1$). If there is only one tour, the assignment of customers to tours is already fixed. Hence, the number of possible solutions is lower. As the customer to tour assignment is fixed, the algorithm has no influence on the compatibility of the customers, i.e. whether they fit well together or not. In contrast, for instances with eight vehicles the relative detour per passenger is even for 20 requests per vehicle (on average, i.e. 160 requests in total) close to 0, indicating reasonable detours that would likely be accepted by consumers in practice.

In a second analysis, we present the maximum relative detour per instance of the second configuration in Fig. 4. A similar effect to Fig. 3 can be observed. In instances with few vehicles, the maximum relative detour is rather large, particularly if the number of customers is also high. On the other hand, for a larger number of vehicles ($k = 8$), the maximum detour is not only lower but also increases

Table 3 Overview of the performance of the second ALNS configuration on all instances and the MIP solved by CPLEX

k	n	ALNS		MIP				
		z	Time [s]	$z^{\dagger}$	Lower bound	Time [s]	#optimal	#feasible
1	5	32.89	0.26	32.89	32.89	20.59	100	100
1	10	118.20	1.12	123.32	0.29	3600.00	0	100
1	15	235.80	3.23	374.04	0.00	3600.00	0	38
1	20	386.30	6.95	1324.75	0.00	3600.00	0	7
2	10	48.61	0.73	52.80	0.86	3597.12	1	100
2	20	168.01	3.91	1542.12	0.00	3,600.00	0	3
2	30	341.39	12.82	–	–	–	–	–
2	40	539.18	30.28	–	–	–	–	–
5	25	63.49	3.62	272.91	0.00	3,600.00	0	1
5	50	195.78	26.24	–	–	–	–	–
5	75	360.12	103.14	–	–	–	–	–
5	100	522.83	276.93	–	–	–	–	–
8	40	47.78	8.82	–	0.00	3600.00	0	0
8	80	138.93	74.82	–	–	–	–	–
8	120	228.96	290.11	–	–	–	–	–
8	160	344.12	851.68	–	–	–	–	–

The MIP was only run for the eight indicated rows. The column #optimal (#feasible) indicates the number of instances that CPLEX reported as optimal (feasible). Each row contains the average of 100 instances

† Average over all feasible instances

more slowly for increasing n (compare the bottom right square to the top right one). Hence, in competition with taxis and private cars, a few further vehicles can make the ride-sharing provider significantly more attractive.

The average relative detour per passenger can further be split into the waiting time $\left(\frac{B_i - r_i}{t_{i,i+n}}\right)$ and the ride time detour $\left(\frac{B_{i+n} - B_i - t_{i,i+n}}{t_{i,i+n}}\right)$. This is visualized in Fig. 5. Particularly for instances with few vehicles, the majority of the objective value is determined by the customers' waiting time. As we increase k, the ratio of waiting to ride time decreases. In a static setting, the pickup time B_i can be communicated to the customer in advance. This can mitigate the negative effect of the high waiting time, as the customers might be able to use the waiting time more effectively than the ride time if they know exactly when they will be picked up.

In Fig. 6, we visualize the development of the ride-sharing effect over different ratios of customers to vehicles. For the instances with a passenger demand of one on the left in Fig. 6, there is little actual ride-sharing when the number of customers per vehicle is low. For $n/k = 5$, for example, almost 80% of the ride time is spent with at most one customer. So the intended effect of reducing the amount of traffic by pooling trips is not actually present. However, the higher pooling rates with a larger

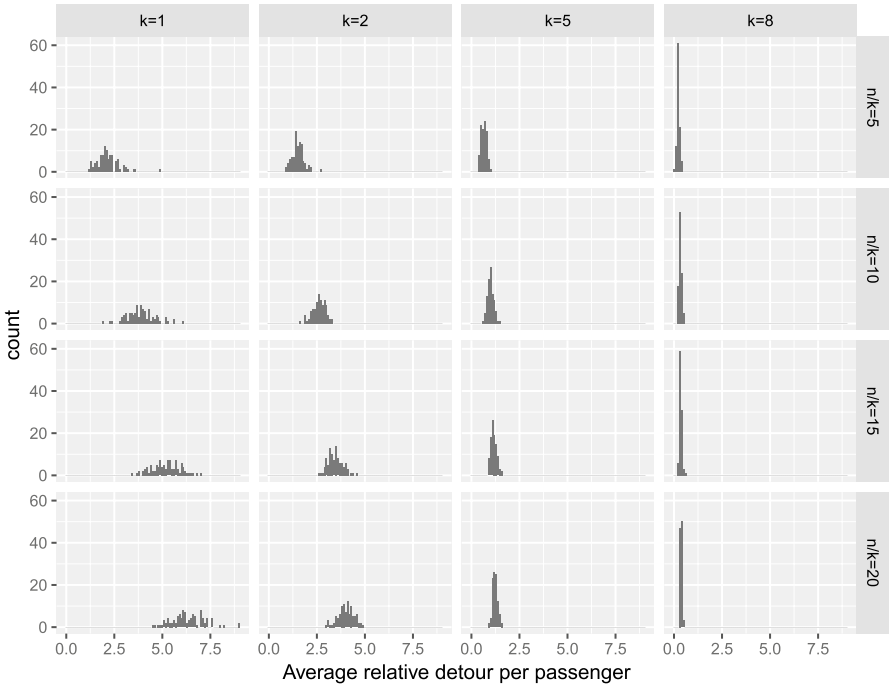

Fig. 3 A histogram of the first part of the objective value per passenger for ALNS split into the 16 instance categories, i.e. the average relative detour in each instance. Each square contains 100 instances

number of customers per vehicle justify the choice of the objective function. The objective function considers exclusively the customer's detour. The results show that a high pooling rate and, hence, a high degree of capacity utilization is accomplished. This means that our objective leads further to good results for the provider on the economic base. In the instances with the exponentially distributed passenger numbers on the right in Fig. 6, the vehicles are more filled on average since even a single request can potentially contain six passengers.

In order to better evaluate the impact of the ride sharing, we ran another test in which we simulated a more traditional taxi environment in which no rides are shared. For this we changed the capacity Q to one for all 800 instances where $q_i = 1$. The results of these instances (when solved with configuration 2 of ALNS) compared to the same instances with the original vehicle capacity of six are shown in Fig. 7. Both the detour per customer as well as the maximum detour (and therefore the objective value) increase when the vehicle capacity is reduced. This effect is more pronounced in the instances with more customers per vehicle which coincides with the findings of Fig. 6 that ride sharing has a larger impact when more customers are present per vehicle.

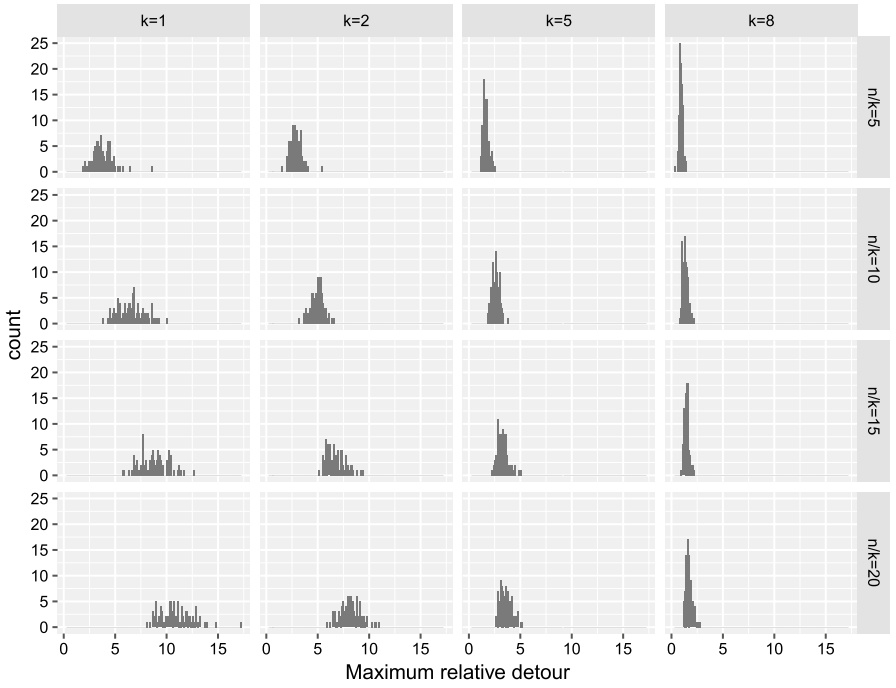

Fig. 4 A histogram of the second part of the objective value for ALNS split into the 16 instance categories, i.e. the maximum relative detour in each instance. Each square contains 100 instances

6.3.2 Results of DP

The restricted DP was run for the 400 instances with $k = 1$. We varied the maximum number of states B for different DP runs to evaluate how many states we need to produce good results in a reasonable time. For the instances with 5, 10, and 15 customers we also provide the optimal solution gained by running the DP without any state restrictions. For the instances with 20 customers this was not possible due to the exponential increase in both time and memory requirements. The results are summarized in Table 4. We also state the result of the second ALNS configuration for comparison. Expectedly, a higher number of states leads to a higher solution quality but also to a higher computational effort. For 5 and 10 customers small state sizes like 5,000 can already find optimal solutions in almost every instance. For 15 and 20 customers the solution quality of the smaller numbers of states in relation to the optimal solution quickly deteriorates. ALNS's solutions for these instances in regard to the objective value are comparable to the restricted DP with 10,000 to 20,000 states for $n = 15$ and 50,000 to 1,000,000 states for $n = 20$. In both cases, ALNS's computation times are similar. These results lead us to believe that there is little benefit to run the DP within the ALNS algorithm as a local optimization procedure since any meaningful improvements to the solution quality of ALNS would require a number of states with too much computational effort. In contrast, the results show that the ALNS finds optimal or near optimal solutions within seconds. Compared

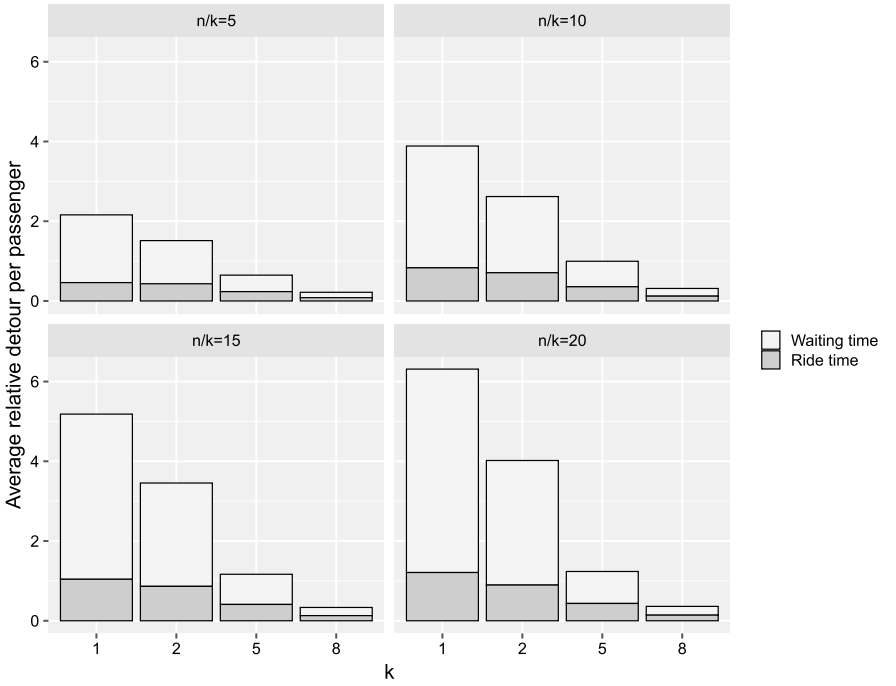

Fig. 5 The first part of the objective value per passenger split into waiting and ride time. Each bar is the average of 100 instances

to the MIP results of Table 3, the DP can clearly beat the MIP for the single tour instances, as the MIP struggles to find optimal solutions for 10 customers or more. In Fig. 8, the average relative difference to the best found solution is given for every state setting (excluding ALNS and the exact DP runs). This further shows that running a restricted DP can still lead to a high solution quality if the number of states is sufficiently high. This is especially beneficial for larger instances where running the exact DP becomes intractable.

7 Conclusion

The paper presents an ALNS and a dynamic programming algorithm for the static dial-a-ride problem with ride and waiting time minimization. Although the dynamic program is restricted to small instances, it can be used with a restricted number of states and shows that for one vehicle the ALNS approach finds near optimal solutions. The results of our computational study relating to the ALNS indicate further that ride-sharing proposals can help to solve the trade-off between individual transport, profitability of the provider, and traffic and pollution reduction. Our results show that the presented approaches lead to solutions with only small detours while generating a significant pooling effect if the number of used

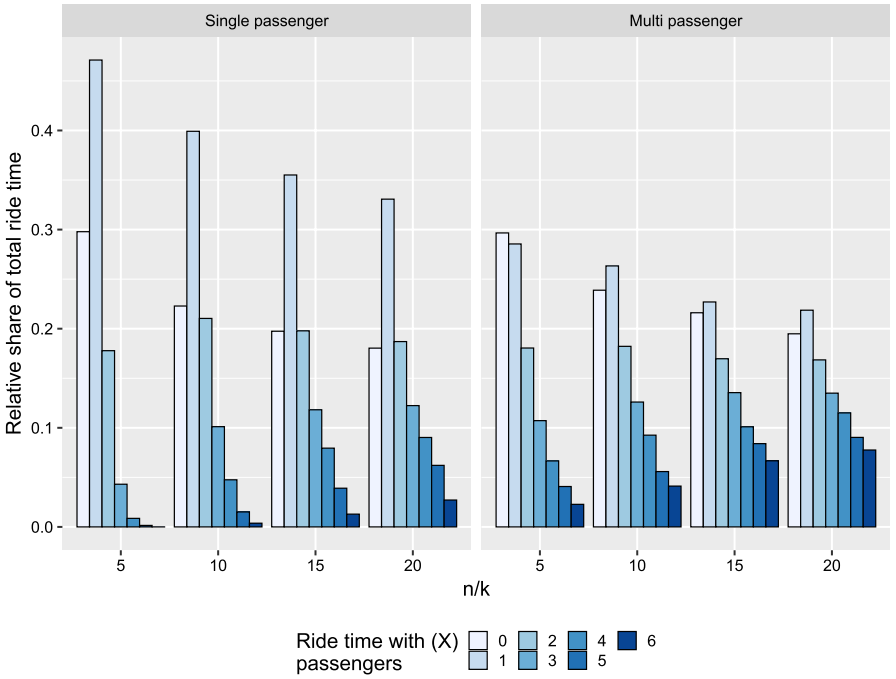

Fig. 6 Development of the occupancy rate for all instances with one passenger per request (left) and all instances with exponentially distributed passenger demand (right). The bars for each instance group add up to 1, each individual bar is the relative share of ride time with that amount of customers. For example, in the single passenger instances with $n/k = 5$, 47% of the total ride time a vehicle has a single passenger

vehicles as well as the number of participating people is not too small. This is a promising result for practical tests and further research.

We studied the static problem setting. Further research is necessary to develop policies for the dynamic setting since customers get an immediate feedback on their request in real applications. Besides, stochastic travel times should be included, as travel times, especially in large cities, are highly dependent on the current traffic situation. Furthermore, the first part of the objective function weights all customers equally while the second part focusses on the maximum relative detour. Our results show that the second part of the objective is important to punish large relative detours. Further research can adapt the second part of the objective to reach more levelled relative detours in the comparison of two customers, e.g. by summing up the pairwise absolute differences of customers' relative detours [compare (Schulz 2021)].

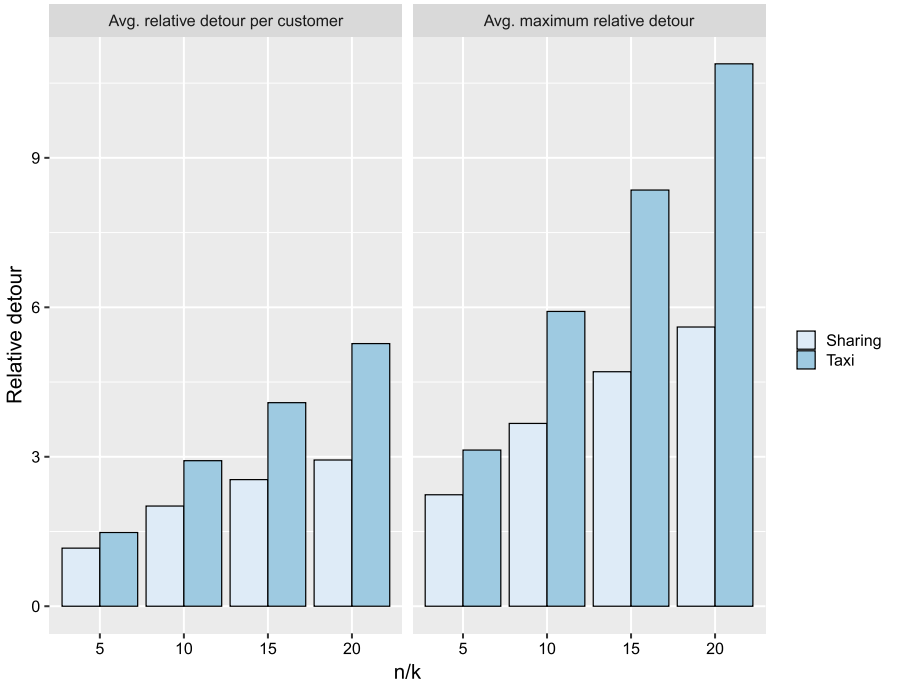

Fig. 7 Comparison of the average detour per customer and the average maximum detour between the instances with $Q = 6$ (Sharing) and $Q = 1$ (Taxi)

Table 4 Summary of objective values and computational times of the DP with different states and ALNS on the instances with one tour

States	$n = 5$		$n = 10$		$n = 15$		$n = 20$	
	z	Time [s]	z	Time [s]	z	Time [s]	z	Time [s]
10	33.82	0.00	140.30	0.00	314.98	0.00	549.61	0.00
25	33.09	0.00	131.01	0.00	290.87	0.00	493.61	0.01
50	32.89	0.00	126.25	0.00	274.87	0.01	472.21	0.02
100	32.89	0.00	122.94	0.01	261.79	0.02	449.16	0.04
500	32.89	0.00	118.80	0.03	246.16	0.09	414.02	0.18
1,000	32.89	0.00	118.36	0.06	241.91	0.19	407.14	0.37
2,500	32.89	0.00	118.31	0.15	239.49	0.46	399.96	0.94
5,000	32.89	0.00	118.18	0.28	237.67	0.93	396.35	1.97
10,000	32.89	0.00	118.18	0.53	236.20	1.93	393.04	4.34
20,000	32.89	0.00	118.18	0.95	235.28	4.13	389.70	9.26
50,000	32.89	0.00	118.18	1.91	235.10	10.67	386.66	23.32
1,000,000	32.89	0.00	118.18	2.73	234.78	172.90	382.79	454.50
Exact	32.89	0.00	118.18	2.66	234.78	1,970.54	–	–
ALNS	32.89	0.26	118.20	1.12	235.80	3.23	386.30	6.95

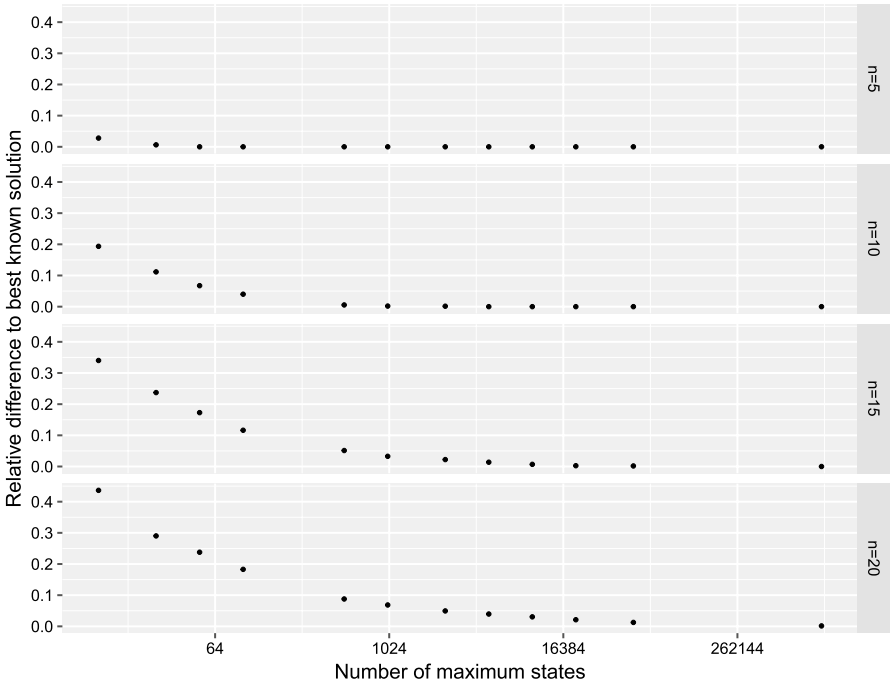

Fig. 8 Relative differences for the restricted DP depending on the maximal number of states to the best known solution for the single tour instances. For $n \in \{5, 10, 15\}$ the best known solution is optimal. The x-axis is scaled logarithmically

A Preliminary study to choose a suitable penalty factor

Determining the value for the penalty factor w is of significant importance. On the one hand, a small w allows for large maximum detours in optimal solutions which are unlikely to be accepted by customers in practice. On the other hand, a high w will eventually sacrifice too much of the average solution quality in order to reduce the maximum detour. When investigating the influence of the penalty factor w on our results, it became apparent that the penalty factor should depend on the number of customers n since the first part of the objective function will also increase with n and the penalty factor needs to scale accordingly. Therefore, we ran a series of tests where we varied the penalty factor w and solved the instances with our ALNS algorithm (set to the tuned parameters returned by *irace* and with every heuristic enabled). The instances were generated as described in Sect. 6.1 with the only difference being that we additionally varied the w between 0, n, $2n$, and $5n$. For each combination of n, k, and w we generated 30 instances, resulting in 1920 instances in total. These instances were not reused in any other tests.

Figure 9 shows the influence of w in our tests. As can be seen on the right in Fig. 9, weighting the maximum detour at all results in a sharp drop of the

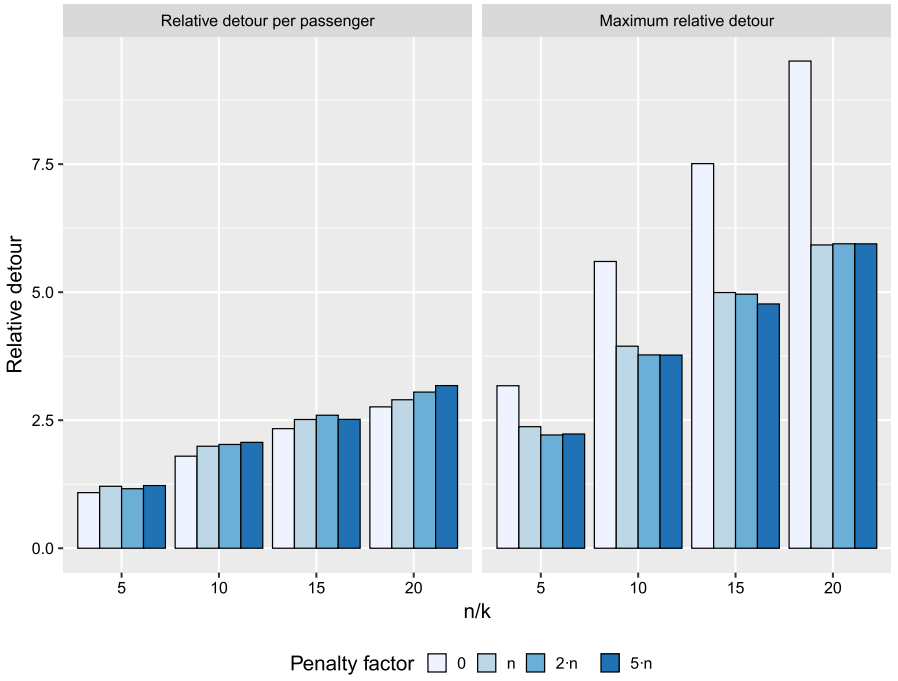

Fig. 9 Development of the average relative detour (left) and the maximum detour (right) when varying the penalty factor w

maximum detour. However, subsequent increases of w only lead to small decreases in the maximum detour. On the other hand, the left plot in Fig. 9 shows that increasing w increases the average detours rather steadily, particularly in the instances with many customers per tour. These results indicate that focusing too much on reducing the maximum detour has adverse effects on the length of the average detour while having little benefit. As a result, a value of $w = n$ seems to strike a good balance between the two objectives.

B Preliminary study to choose good configurations

Prior research has shown that a preliminary selection of heuristics for an ALNS implementation is of high importance (François et al. 2016). This means that it might be advantageous to only enable a subset of the heuristics. One simple approach to determine a good configuration of heuristics is to repeat the following procedure for every heuristic: Disable the heuristic and examine whether the algorithm's performance improves. However, this approach clearly does not take into account the interaction effects between the different heuristics. If we have an initial selection of heuristics that focus too much on diversification, then the individual tests might lead us to remove all the heuristics that diversify. This could result in a configuration with too little diversification. Furthermore, since we examine a new problem,

previous results from the literature in regard to good heuristic configurations are of limited use. Therefore, we decided to run a set of tests to evaluate every possible configuration. As we utilize seven removal and six insertion ($g = 2$ and $g = 3$ for insertion heuristic 4.) heuristics and we always require at least one removal and one insertion heuristic to be enabled, we need to test $2^{13} - 2^7 - 2^6 + 1 = 8001$ configurations. The test bed consists of 24 instances with varied sizes and $w = n$. These instances were only used for the preliminary testing, not for the computational study. In order to find "good" configurations, we examined every instance and calculated the squared relative difference of the objective values from each configuration to the configuration that performed best on the instance. A histogram of the 8001 configurations and their mean squared relative differences can be seen in Fig. 10. The figure shows a large cluster of configurations that are all reasonably good. However, in order to run tests on more instances and increase the reliability of the results, we needed to select a small subset of configurations for further tests. Therefore, we selected the 100 configurations with the lowest mean squared relative difference for more in-depth tests. This cut-off is also shown in the figure.

In order to further examine these 100 configurations, we generated another, larger test bed of instances with different sizes (different from the one before and the one in the computational study). We varied the number of vehicles between 2, 5, and 8 and chose the number of customers such that the number of customers per

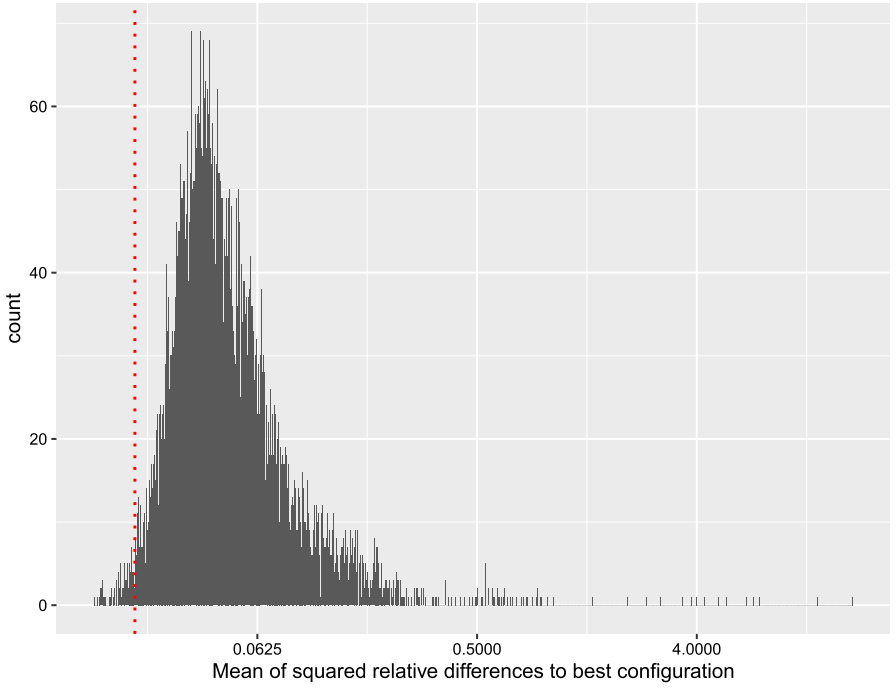

Fig. 10 Histogram of the 8001 configurations' mean of the squared relative differences to the corresponding best known solution. There are 100 configurations left of the dotted line which were then evaluated in further tests. The x-axis is scaled logarithmically to better display some of the outliers

vehicle (i.e. n/k) equals 5, 10, and 15. This results in nine parameter configurations for which we drew 50 instances each, leading to 450 instances in total. Each configuration was then run on each instance. When evaluating the configurations, we focused on two criteria: The first is the mean objective value of each configuration over all instances. The second is the aforementioned mean squared relative difference of each configuration to the best known solution. This second criterion punishes outliers and, therefore, rewards configurations that perform consistently well over all instances. A visualization can be seen in Fig. 11. Each dot represents the average results over all 450 instances of one configuration. Further, the regression line is included which shows that both evaluation criteria (mean and squared relative difference) have a strong positive correlation, which means that configurations with a low mean objective value perform consistently well over all instances. No configuration was always the best on every instance, which is why the squared relative difference is always positive. The two configurations that performed best are the two pareto optimal configurations marked in the figure by numbers 1 and 2. The utilized heuristics of these two pareto optimal configurations can be seen in Table 2. Neither configuration uses *Cluster removal* nor *Regret 2 insertion* though in the latter case this can likely be explained by the use of the *Regret 3 insertion*. Interestingly, configuration 1 uses all heuristics that configuration 2 uses with the exception of *Related removal distance*. The utilized insertion heuristics for both configurations

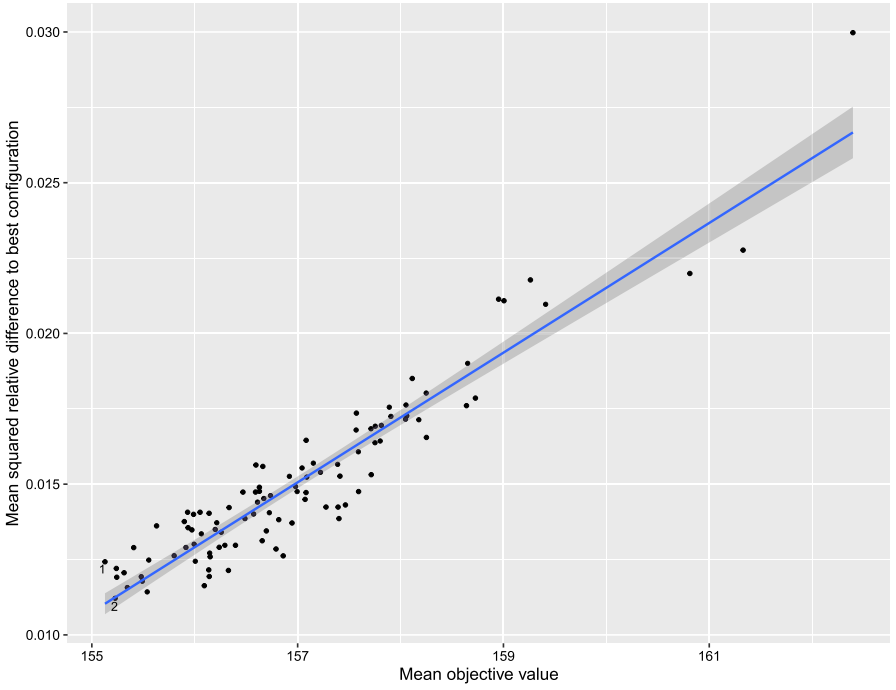

Fig. 11 Evaluation of the 100 configurations according to two criteria. Each point represents a configuration. The two named configurations are the two configurations that are pareto optimal for both criteria

only differ in the use of *Basic greedy insertion* which is likely covered by the other randomized greedy insertion variants in configuration 2.

Funding Open Access funding enabled and organized by Projekt DEAL. 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: https://doi.org/10.25592/uhh-fdm.9610.

Declarations

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

References

Atahran A, Lenté C, T'kindt V, (2014) A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. J Multi-Criteria Decis Anal 21(5-6):279–298

Baldacci R, Bartolini E, Mingozzi A (2011) An exact algorithm for the pickup and delivery problem with time windows. Oper Res 59(2):414–426

Baugh JW, Kakivaya GKR, Stone JR (1998) Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. Eng Optim 30(2):91–123

Bellman R (1962) Dynamic programming treatment of the travelling salesman problem. J ACM 9(1):61–63

Bischoff J, Maciejewski M (2016) Simulation of city-wide replacement of private cars with autonomous taxis in Berlin. Procedia Comput Sci 83:237–244

Chevrier R, Liefooghe A, Jourdan L, Dhaenens C (2012) Solving a dial-a-ride problem with a hybrid multi-objective evolutionary approach: application to demand responsive transport. Appl Soft Comput 12(4):1247–1258

Cordeau JF (2006) A branch-and-cut algorithm for the dial-a-ride problem. Oper Res 54(3):573–586

Cordeau JF, Laporte G (2003) A tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transp Res Part B Methodol 37(6):579–594

Cordeau JF, Laporte G (2007) The dial-a-ride problem: models and algorithms. Ann Oper Res 153(1):29–46

Cubillos C, Urra E, Rodríguez N (2014) Application of genetic algorithms for the darptw problem. Int J Comput Commun Control 4(2):127

Detti P, Papalini F, de Lara GZM (2017) A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega 70:1–14

Diana M, Dessouky MM (2004) 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

François V, Arda Y, Crama Y, Laporte G (2016) Large neighborhood search for multi-trip vehicle routing. Eur J Oper Res 255(2):422–441
Gschwind T, Drexl M (2019) Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transp Sci 53(2):480–491
Gschwind T, Irnich S (2015) Effective handling of dynamic time windows and its applications to solving the dial-a-ride problem. Transp Sci 49(2):335–354
Heilporn G, Cordeau JF, Laporte G (2011) An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays. Discret Appl Math 159(9):883–895
Held M, Karp RM (1962) A dynamic programming approach to sequencing problems. J Soc Ind Appl Math 10(1):196–210
Hemmelmayr VC, Cordeau JF, Crainic TG (2012) An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Comput Oper Res 39:3215–3228
Ho SC, Szeto WY, Kuo YH, Leung JMY, Petering M, Tou TWH (2018) A survey of dial-a-ride problems: literature review and recent developments. Transp Res Part B Methodol 111:395–421
Hunsaker B, Savelsbergh M (2002) Efficient feasibility testing for dial-a-ride problems. Oper Res Lett 30(3):169–173
ioki GmbH (2020) ioki – inspiring smart mobility. https://ioki.com/en/home/. Accessed Sep 27, 2021
Jaw JJ, Odoni AR, Psaraftis HN, Wilson NH (1986) 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
Jorgensen RM, Larsen J, Bergvinsdottir KB (2007) Solving the dial-a-ride problem using genetic algorithms. J Oper Res Soc 58(10):1321–1331
Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Am Math Soc 7(1):48–50
Lees-Miller JD, Wilson RE (2012) Proactive empty vehicle redistribution for personal rapid transit and taxis. Transp Plan Technol 35(1):17–30
Lehuédé F, Masson R, Parragh SN, Péton O, Tricoire F (2014) A multi-criteria large neighbourhood search for the transportation of disabled people. J Oper Res Soc 65(7):983–1000
López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, Birattari M, Stützle T (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 3:43–58
Malandraki C, Dial RB (1996) A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. Eur J Oper Res 90(1):45–55
GmbH MOIA (2020) Our reunion begins here. https://www.moia.io/en. Accessed Sep 27, 2021
Molenbruch Y, Braekers K, Caris A (2017a) Benefits of horizontal cooperation in dial-a-ride services. Transp Res Part E 107:97–119
Molenbruch Y, Braekers K, Caris A, den Berghe GV (2017b) Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation. Comput Oper Res 77:58–71
Muelas S, LaTorre A, Peña JM (2013) A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. Expert Syst Appl 40:5516–5531
Nie W (2000) Waiting: integrating social and psychological perspectives in operations management. Omega 28(6):611–629
Oxley P (1980) Dial/a/ride: a review. Transp Plan Technol 6:141–148
Parragh SN, Doerner KF, Hartl RF (2008) A survey on pickup and delivery models: Part II: transportation between pickup and delivery locations. J Betriebswirtschaft 58(2):81–117
Parragh SN, Doerner KF, Hartl RF, Gandibleux X (2009) A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks 54(4):227–242
Parragh SN, Doerner KF, Hartl RF (2010) Variable neighborhood search for the dial-a-ride problem. Comput Oper Res 37(6):1129–1138
Parragh SN, Pinho de Sousa J, Almada-Lobo B (2015) The dial-a-ride problem with split requests and profits. Transp Sci 49(2):311–334
Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34:2403–2435
Psaraftis HN (1980) A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transp Sci 14(2):130–154
Psaraftis HN (1983) An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transp Sci 17(3):351–357

Reinhardt LB, Clausen T, Pisinger D (2013) Synchronized dial-a-ride transportation of disabled passengers at airports. Eur J Oper Res 225(1):106–117
Ritzinger U, Puchinger J, Hartl RF (2016) Dynamic programming based metaheuristics for the dial-a-ride problem. Ann Oper Res 236(2):341–358
Ropke S, Pisinger D (2006a) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 40(4):455–472
Ropke S, Pisinger D (2006b) A unified heuristic for a large class of vehicle routing problems with backhauls. Eur J Oper Res 171:750–775
Ropke S, Cordeau JF, Laporte G (2007) Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 49(4):258–272
Schilde M, Doerner KF, Hartl RF (2011) Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. Comput Oper Res 38(12):1719–1730
Schulz A (2021) The balanced maximally diverse grouping problem with block constraints. Eur J Oper Res 294(1):42–53
Sexton TR, Bodin LD (1985) Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. Transp Sci 19(4):378–410
Shaw P (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. Department of Computer Science, University of Strathclyde, Scotland, Tech. rep
Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget J-F (1998) Principles and practice of constraint programming - CP98, LNCS 1520. Springer, pp 417–431
Tang J, Kong Y, Lau H, Ip AW (2010) A note on efficient feasibility testing for dial-a-ride problems. Oper Res Lett 38(5):405–407
Technologies Uber, Inc (2020) The history of Uber. https://www.uber.com/en-DE/newsroom/history/. Accessed Sep 27, 2021
Wilson NH, Borndörfer R, Grötschel M, Klostermeier F, Küttner C (eds) (1999) Telebus Berlin: vehicle scheduling in a dial-a-ride system: computer-aided transit scheduling. Springer, Berlin