Ride-sharing with flexible participants: a metaheuristic approach for large-scale problems

Pieter Smet*

*KU Leuven, Department of Computer Science, CODES & imec, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
E-mail: pieter.smet@cs.kuleuven.be*

Received DD MMMM YYYY; received in revised form DD MMMM YYYY; accepted DD MMMM YYYY

Abstract

Severe traffic congestion and growing ecological consciousness has led to the rise of alternative transportation systems. Ride-sharing is one such alternative in which drivers and passengers with similar time schedules and travel plans are matched. For this service to be effective, a large number of users is required to increase the probability of finding suitable travel partners. The present paper proposes a late acceptance metaheuristic to decide which users act as drivers and to construct their routes. The underlying optimization model allows passengers to walk to/from alternative pickup/drop-off locations so as to further exploit user flexibility. A computational study quantifies the impact of different types of participant flexibility on CO₂ emissions. These insights can inform and support policymakers in organizing effective ride-sharing systems.

Keywords: ride-sharing; carpooling; late acceptance hill climbing; metaheuristic

1. Introduction

Traffic congestion is a critical issue in urban areas and on connecting highways. Figure 1 shows the number of hours spent in congestion in the twenty most congested cities in Europe. On average, commuters in these urban areas spend 25% of their trip in congestion during peak hours. In Brussels, Belgium, the average driver spends 39 hours in traffic jams annually. Even worse, there is no indication these statistics will improve in the near future (European Commission, 2015). This trend, combined with a growing consciousness of the ecological consequences of excessive car usage, has motivated commuters to find alternative modes of transportation to reach their destination. Ride-sharing is an emerging transportation alternative wherein users with similar time and travel requirements share a car, thereby directly reducing the number of cars on the road (Furuhata et al., 2013). Technologies such as geographic information systems, global positioning systems and smartphones have facilitated implementation and accessibility

* Author to whom all correspondence should be addressed (e-mail: pieter.smet@cs.kuleuven.be).

of ride-sharing (Chan and Shaheen, 2012).

Fig. 1: Time spent in congestion in European urban areas (INRIX, 2017)

Motivated by the increasing interest in ride-sharing services, this paper addresses an operational optimization problem which is characterized by two main decisions: (i) determining which users act as drivers and (ii) constructing driver routes. A route is defined as a path beginning from a driver's origin location and ending at their destination. Along the route, passengers may be picked up and dropped off at their origin and destination locations as long as a number of constraints concerning the route's structure are respected. Passengers may be willing to walk a short distance to an alternative pickup location (or from an alternative drop-off location) which may potentially improve overall solution quality.

Figure 2 shows an example of two feasible routes in which users walk to/from alternative pickup/drop-off locations as indicated by the dashed arrows. The first route starts at the origin location of its driver, user 1. Before arriving at the route end point, that is, the destination location of user 1, user 2 is picked up and subsequently dropped off at his destination. The route of user 3 as driver has a more complex structure and makes use of alternative pickup and drop-off locations. The first stop is the origin location of user 4. This location also serves as an alternative pickup location for user 5, who also gets in the car at this point. Next, user 4 is dropped off at his destination. The route continues to pick up user 6 who has the same destination as user 5. The route ends at the driver's destination which serves as an alternative drop-off location for users 5 and 6 who continue on foot to their destination location.

An essential characteristic of the proposed model is driver flexibility, which assumes that the role of a user is not fixed a priori. A subset of users, called *flexible drivers*, own a car but are also willing to be a passenger in another car. Combined with alternative service locations and time windows which allow for detours, the proposed model enables to maximally exploit flexibility of participants in ride-sharing systems. The underlying optimization problem may be considered a generalization of the classical Vehicle Routing Problem, which is known to be NP-hard in the strong sense.

Ride-sharing is considered here in an offline setting, that is, all requests that need to be assigned and scheduled are known at the time the problem is solved. In practice, this assumption may not always hold as requests arrive in a dynamic manner and only become known over time. In addition to optimizing

Fig. 2: Example of feasible routes which includes users walking to/from alternative pickup/drop-off locations.

static operational ride-sharing problems, we position the current contribution also as a tool to inform policymakers responsible for making strategic decisions regarding the design of ride-sharing services. Trade-offs may be identified based on simulations which allow to evaluate whether a given scenario may benefit from ride-sharing and which measures should be taken to increase the impact of ride-sharing.

The present paper's main contributions may be summarized as follows:

- A general model is developed for a complex ride-sharing system which addresses both assignment and sequencing decisions. The proposed model combines various characteristics of existing models, including different types of flexibility of the system's participants.
- A late acceptance metaheuristic is introduced for optimizing large-scale problem instances. The proposed solution approach is evaluated using a dataset consisting of both real-world and computer-generated problem instances representing ride-sharing under varying conditions.
- The impact of relaxing problem characteristics is analyzed in a computational study involving different simulations which policymakers may consider relevant. These experiments demonstrate the trade-off between cost reduction and the degree of participant flexibility.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work previously published in the academic literature. Section 3 presents a formal definition of the problem. Section 4 discusses the proposed metaheuristic solution approach and its different components. Section 5 analyzes a series of computational experiments to gain insights into the algorithm's performance and the effect of various parameters and problem properties. Section 6 quantifies the trade-off between cost reduction and the degree of participant flexibility. Finally, Section 7 concludes the paper and outlines directions for future research.

2. Related work

2.1. Ride-sharing and carpooling

The academic literature addresses different types of ride-sharing. One such form is carpooling, which is typically defined as either a daily or a long-term problem (Maniezzo et al., 2004). The long-term problem seeks to design user pools in which each user will, in turn, act as driver on different days. The objective is to maximize the size of the user pools, thereby minimizing the number of required vehicles. By contrast, the daily problem assigns users to drivers who are available on that day, subject to operational constraints such as time windows and vehicle capacity. Due to the computational complexity of the daily and long-term problems, they are typically solved separately, that is, a solution for the long-term problem defines the input for solving the daily problem.

The daily ride-sharing problem is further categorized based on the users' origin and destination locations. For the many-to-one ($n-1$) problems, in which users share the same destination, both exact approaches and heuristics have been proposed (Baldacci et al., 2004; Huang et al., 2016). Bruck et al. (2017) investigate a case study of an Italian hospital in which users are allowed to drive to intermediate pickup points and continue with a single car from there on. Using a local search heuristic, they report a reduction in CO2 emissions of up to 40%. Kaan and Olinick (2013) consider a similar problem in which users agree to meet at a target location from which they travel to a shared destination in a rental van.

The present paper considers a many-to-many ($n-m$) problem in which users from different origin locations may also have different destinations. Wolfier Calvo et al. (2004) describe a decision support system for organizing carpooling services. The core of their system is a local search algorithm for assigning passengers to drivers and constructing the drivers' routes. Computational results are reported for problem instances with up to 600 users. Chou et al. (2016) emphasize the multi-objective nature of carpooling by considering several optimization criteria including travel time, number of unserved users, preferences and waiting time. Problem instances with up to 300 users are addressed using a particle swarm optimization heuristic. Yan and Chen (2011b) present a Lagrangian relaxation algorithm for solving a multi-commodity network flow formulation, the size of which is strongly determined by the problem's time granularity. The scheduling horizon is discretized in time intervals of 15 minutes to ensure acceptable computation times for problem instances with up to 2200 users. This model is extended by Yan and Chen (2011a) to include pre-matching information obtained from previous carpooling solutions to ensure consistency among matched users and the drivers' routes.

Most models from the academic literature assume travel times to be deterministic, while in practice, these times are stochastic variables due to, for example, traffic congestion during peak hours (Yan et al., 2014). Optimal ride-sharing solutions obtained by deterministic models may become suboptimal or even infeasible when considering stochastic travel times. Long et al. (2018) address a static ride-sharing problem with travel time uncertainty under the single-driver single-rider assumption. This assumption ensures each driver only stops once for a pickup and drop-off, thereby eliminating sequencing decisions from their optimization model. A computational study analyzes the impact of different model properties on the system's performance.

Agatz et al. (2012) propose a bipartite matching model for a single-driver single-rider problem in a dynamic setting in which trip requests arrive to the system over time given that users and drivers send their information to the platform in real-time. A rolling horizon approach is applied in which time is

discretized into multiple planning horizons such that the dynamic problem may be solved as a sequence of static problems. This single-driver single-rider model is further analyzed by Stiglic et al. (2015, 2016) who investigate the effect of different types of scheduling flexibility and the presence of meeting points, respectively.

Table 1 compares the present paper with previously published models for ride-sharing and carpooling problems. The proposed model optimizes a commonly used objective function which minimizes travel costs and the number of unassigned users. Three different types of participant flexibility from previous studies are combined: time windows on user pickup and drop-off, flexible participant role assignment and alternative pickup or drop-off locations. The two primary decisions in the proposed model concern (i) partitioning users into drivers and passengers, and (ii) sequencing stops to construct driver routes.

	Cardinality	Time windows	Flexible drivers	Alternative locations	Type	Objectives
Bruck et al. (2017)	n-1	no	no	yes	static	CO2
Kaan and Olinick (2013)	n-1	yes	no	yes	static	TC/UAS
Baldacci et al. (2004)	n-1	yes	no	no	static	TC/UAS
Huang et al. (2016)	n-1	yes	yes	no	static	TC
Chou et al. (2016)	n-m	no	no	no	static	TC/UAS/PRF/WAIT
Wolfler Calvo et al. (2004)	n-m	yes	no	no	static	TC/UAS
Yan and Chen (2011a)	n-m	yes	no	no	static	TC/UAS
Yan and Chen (2011b)	n-m	yes	no	no	static	TC/UAS
Lin et al. (2012)	n-m	yes	no	no	static	TC/WAIT/DET
Stiglic et al. (2016) ¹	n-m	yes	no	yes	dynamic	TC/UAS
Agatz et al. (2012) ¹	n-m	yes	yes	no	dynamic	TC/UAS
Yan et al. (2014) ²	n-m	yes	yes	no	static	TC/UAS
Long et al. (2018) ^{1,2}	n-m	yes	yes	no	static	TC/UAS,TW
This paper	n-m	yes	yes	yes	static	TC/UAS

¹ Assumes the single-driver single-rider model in which a driver is restricted to at most one additional pickup and drop-off stop.

² Includes stochastic travel times.

Table 1: Comparison of models from the academic literature. The following abbreviations are used: CO2: CO2 emissions, TC: travel costs, UAS: unassigned users, PRF: driver/passenger preferences, WAIT: waiting time, TW: time window violations, DET: detour time.

2.2. Other vehicle routing problems

Ride-sharing bears similarities with the well-known Pickup and Delivery Problem (PDP) and its special case, the Dial-A-Ride Problem (DARP). Each user within the DARP is associated with a transportation request from an origin to a destination location. The goal is to construct a number of vehicle routes starting from a central depot such that all users are picked up and dropped off while minimizing the total travel distance, subject to a variety of side constraints (Cordeau and Laporte, 2003). Many DARP constraints and objectives are also relevant in the context of ride-sharing. For example, users typically express the earliest time they can leave their origin location and the latest time by which they must have

arrived at their destination, thereby defining left (right) unbounded time windows at each destination (origin) location and the maximum ride time of each user. Note that in practice, these constraints are often not strict and may thus be modeled as soft constraints by minimizing, for example, lateness in the objective function (Jørgensen et al., 2007). If the service provider has no obligation to service all users, hard time windows lead to avoiding user rejection becoming an additional objective (Parragh et al., 2015).

The considered ride-sharing problem with flexible participants generalizes the PDP and DARP in two distinct ways. Firstly, by assuming flexible drivers, the role of each user is not determined a priori. As a result, an additional decision is required to determine the number of vehicles and the best start and end locations of these vehicles. Note that there is no objective term explicitly related to the number of vehicles used in a solution. Secondly, allowing users to consider alternative pickup and drop-off locations also presents a generalization of the PDP and DARP which typically specify fixed pickup and delivery locations for each request.

3. Problem definition

Let $U = \{1, \dots, |U|\}$ be the set of users, with $D \subseteq U$ the subset of users who may act as driver. Let $\bar{D} \subseteq D$ be the subset of obliged drivers who can only act as driver and never as passenger. This defines a set of flexible drivers $\tilde{D} = D \setminus \bar{D}$ who may be either driver or passenger. Each user $u \in U$ is characterized by an earliest possible departure time e_u from when they may leave their origin location, a latest possible arrival time l_u by which they should arrive at their destination and a maximum time m_u they are willing to walk to or from an alternative pickup or drop-off location. For each driver $k \in D$, the number of available passenger seats q_k is given. A service time P is associated with each pickup and drop-off to represent the time required by passengers for getting in or out of a car. A fixed cost C represents the cost of leaving a user unassigned. While there is no explicit constraint on route length, the maximum detour time is restricted by the definition of the earliest departure and latest arrival times. It is assumed that users' time windows are set carefully: wide time windows may lead to one driver acting as a taxi service picking up and dropping off many other users before arriving at their own destination.

Let $G = (V, A)$ be a complete directed graph with V the set of nodes representing all pickup and drop-off locations, and the set of arcs $A = \{(i, j) : i, j \in V, i \neq j\}$. The nodes associated with the origin and destination locations of user u are denoted by $h(u)$ and $w(u)$, respectively. Three parameters are associated with each arc $(i, j) \in A$: the distance d_{ij} , the travel time by car t_{ij} and the travel time on foot w_{ij} . Based on the maximum walking time m_u , the sets of feasible alternative pickup locations $O_u^p = \{j : j \in V, w_{h(u)j} \leq m_u\}$ and alternative drop-off locations $O_u^d = \{i : i \in V, w_{iw(u)} \leq m_u\}$ may be computed. When a user u is picked up at an alternative pickup location $o' \in O_u^p$, the earliest possible departure time at o' becomes $e_u + w_{h(u)o'}$. Similarly, when an alternative drop-off location $o' \in O_u^d$ is selected for user u , the latest arrival time at o' is $l_u - w_{o'w(u)}$.

The goal is to generate routes in which users are picked up and dropped off at feasible locations by drivers respecting (i) the users' time windows and (ii) the drivers' car capacity. The objective is to assign as many users as possible while minimizing the total distance traveled. A weighted sum is used to optimize these two objectives simultaneously. Appendix A introduces an integer programming formulation of the problem.

4. Late acceptance hill climbing

The considered problem is often solved multiple times per day. For example, in a system organizing ride-sharing for employees commuting to and from work, the problem is solved twice per day: once in the morning and again in the evening. Computational experiments, detailed in Section 5, demonstrate how state-of-the art integer programming solvers do not offer a viable solution approach for instances with varying problem structure, given the restriction on computation time. Therefore, late acceptance hill climbing (LAHC) is proposed as a metaheuristic approach. Previous research has proven this method to be effective in solving other challenging combinatorial optimization problems which involve assignment and sequencing decisions (Fonseca et al., 2016; Van Den Dooren et al., 2017; Bolaji et al., 2018).

LAHC differs from the classic hill climbing algorithm in its acceptance criterion: rather than comparing the neighboring solution to the previous solution, it is compared against the solution of several iterations before (Burke and Bykov, 2017). LAHC uses a single parameter, the list length λ , which determines how many previous solution costs are stored. In general, large values of λ result in better solutions but require more time to converge.

Algorithm 1 outlines LAHC in pseudocode. Let $f(S)$ be a function that returns the objective value of a solution S . An initial solution S_0 is generated using a fast constructive heuristic which designates all car owners as drivers. Each passenger is then assigned to a random driver such that all constraints are satisfied. If no such assignment is found, the passenger remains unassigned. LAHC starts by initializing the list of previous objective values $\vec{p}_k$ with the cost of the initial solution $f(S_0)$ (line 4). In its main loop, the algorithm samples one feasible neighboring solution S' from a randomly selected neighborhood N (lines 6-7). Based on the new objective value and the values stored in the list $\vec{p}_k$, S' is accepted or rejected and the best solution S^* is updated (lines 8-14). At the end of each iteration, the list is updated (line 15). When the time limit is reached, the best found solution S^* is returned.

4.1. Solution representation and feasibility testing

The proposed LAHC implementation operates on a direct solution representation defined by two data structures: a set Q of unassigned users and, for each driver $k \in D$, an ordered list R_k of tuples representing the different stops in the driver's route. Each tuple consists of a user u , a location o and an indicator TYPE which denotes whether user u is to be picked up or dropped off. Formally, a route R_k for driver k is defined as shown in Equation (1).

$$R_k = \langle (u_1, o_1, \text{TYPE}_1), (u_2, o_2, \text{TYPE}_2), \dots, (u_{|R_k|}, o_{|R_k|}, \text{TYPE}_{|R_k|}) \rangle \quad (1)$$

For each user u there are two stops (u, o, PICKUP) and $(u, o', \text{DROPOFF})$ which represent their pickup and drop-off. These two stops always appear in the same route R_k . Moreover, each route R_k is always precedence-feasible in the sense that the pickup stop of user u will always appear before the drop-off stop. A route always begins and ends with a stop representing the driver's pickup and drop-off, respectively.

To determine whether a route respects the time windows and capacity constraints, the feasibility test outlined in Algorithm 2 is used. Each stop is scheduled as early as possible so as to avoid arriving after the latest arrival time. Let t_n be the time at which stop $(u_n, o_n, \text{TYPE}_n)$ is scheduled. This procedure

Algorithm 1 Late acceptance hill climbing**Input:** $S_0, f(S), \lambda$ **Output:** S^*

```

1:  $S \leftarrow S_0$                                  $\{S \text{ denotes the current solution}\}$ 
2:  $S^* \leftarrow S_0$                              $\{S^* \text{ denotes the the best solution}\}$ 
3:  $i \leftarrow 0$                                  $\{i \text{ maintains the number of iterations}\}$ 
4:  $\vec{p}_k \leftarrow f(S_0), \forall k \in \{1, \dots, \lambda\}$      $\{\text{Initialize late acceptance list}\}$ 
5: while time limit not exceeded do
6:    $N \leftarrow \text{SelectNeighborhood}()$ 
7:    $S' \leftarrow N(S)$                              $\{N(S) \text{ selects a neighboring solution of } S\}$ 
8:    $v \leftarrow i \bmod \lambda$ 
9:   if  $f(S') \leq \vec{p}_v$  then
10:     $S \leftarrow S'$ 
11:    if  $f(S) < f(S^*)$  then
12:       $S^* \leftarrow S$ 
13:    end if
14:  end if
15:   $\vec{p}_v \leftarrow f(S)$ 
16:   $i \leftarrow i + 1$ 
17: end while
18: return  $S^*$ 

```

runs in linear time $O(|R_k|)$. Note that the time window constraint can also be handled in constant time by making use of Forward Time Slack (Savelsbergh, 1992). The capacity constraint is verified by updating the number of users in the car (the *load*) at each stop, and verifying after each pickup stop if the maximum car capacity q_k is not exceeded.

4.2. Neighborhoods

In each LAHC iteration, one neighborhood operator is randomly selected. Rather than fully exploring the selected neighborhood, a single neighboring solution is randomly selected. The following five intra-route neighborhoods are explored in the proposed algorithm:

- **Change order (CO)**: given a route R_k , move the pickup and drop-off stops of a given user u to different positions within R_k .
- **Change location (CL)**: given a tuple (u, o, TYPE) in a route R_k , change the location o to an alternative location $o' \in O_u^p$ or $o' \in O_u^d$ depending on the stop's TYPE.
- **Delete user (DU)**: given a route R_k , remove the pickup and drop-off stops of a given user u and update the set of unassigned users accordingly $Q \cup \{u\}$.
- **Add user (AU)**: given an unassigned user $u \in Q$, assign its pickup and drop-off stops to a randomly selected route R_k .
- **Replace user (RU)**: given a route R_k , remove the pickup and drop-off stops of a given user u such

Algorithm 2 Feasibility testing of route R_k **Input:** List of stops R_k **Output:** Boolean indicating feasibility of R_k

```

1:  $load \leftarrow 0$  {Keep track of number of users in car}
2: for  $(u_n, o_n, TYPE_n) \in R_k$  do
3:   if  $TYPE_n = \text{PICKUP}$  then
4:      $t_n \leftarrow \max(e_{u_n}, t_{n-1} + t_{o_{n-1}, o_n}) + P$ 
5:      $load \leftarrow load + 1$ 
6:     if  $load > q_k$  then
7:       return false {Violation of capacity constraint}
8:     end if
9:   else
10:     $t_n \leftarrow t_{n-1} + t_{o_{n-1}, o_n}$ 
11:     $load \leftarrow load - 1$ 
12:    if  $t_n > l_{u_n}$  then
13:      return false {Violation of time window constraint}
14:    end if
15:     $t_n \leftarrow t_n + P$ 
16:  end if
17: end for
18: return true

```

that $Q \cup \{u\}$ and insert the pickup and drop-off stops of an unassigned user $u' \in Q$.

In addition, the following two inter-route neighborhoods are explored:

- **Move user (MU):** given two routes R_k and $R_{k'}$ with $k \neq k'$, move the pickup and drop-off stops of a given user u from R_k to random positions in $R_{k'}$.
- **Swap users (SU):** given two routes R_k and $R_{k'}$ with $k \neq k'$, swap the pickup and drop-off stops of two given users u and u' to random different positions within $R_{k'}$ and R_k .

Delta evaluation is used such that routes which have not been modified are not re-evaluated. As neighborhood operators only modify up to two routes, this allows for a significant speedup compared to re-evaluating a full solution. After executing MU or SU both R_k and $R_{k'}$ must be re-evaluated, while after executing CO, CL, DU, AU or RU only the objective value of R_k must be recomputed. Moreover, since a route is typically composed of a limited number of stops, delta evaluation enables to increase the number of iterations per second by several orders of magnitude.

5. Computational study

5.1. Instance generation and experimental setup

Two datasets are employed in the computational study. Class A represents ride-sharing for short distances in an urban area, while class B instances are based on real-world data provided by a carpooling company describing the morning commute of residents of a large city. The users' time windows were set based on the methodology proposed by Stiglic et al. (2016). The earliest departure time e_u of user u is randomly sampled from a normal distribution with standard deviation $\sigma = 30$ minutes, truncated at 2σ , resulting in a travel peak around the distribution's mean. Travel distance is calculated by taking the direct distance between origin and destination and increasing it with 30% to approximate real distances on the road network. Based on the average driving speed, which depends on the problem class, the latest arrival time l_u is calculated starting from the earliest departure time. It is assumed that each driver is willing to extend their trip duration by 25% to pick up and drop off other users.

In all instances, 40% of the users were selected as obliged drivers and 40% as flexible drivers. The remaining 20% may only act as passengers. The maximum walking time m_u of user u was sampled from a uniform distribution between 0 and 15 minutes with a granularity of five minutes. Average walking speed was fixed to 4.8 kilometers per hour. The service time P was set to two minutes.

5.1.1. Class A: urban area instances

The first class of instances was generated according to the procedure described by Stiglic et al. (2016). All origin and destination locations were uniformly randomly generated in an area of 10 kilometers by 10 kilometers. Trips shorter than three kilometers were rejected. An average driving speed of 30 kilometers per hour was considered. Car capacity was randomly sampled from a uniform distribution to be between one and three, excluding the driver's seat. In total 50000 trips were generated, from which 15 instances were derived by randomly sampling trips, resulting in instances with a varying number of users from 200 to 3000¹.

5.1.2. Class B: commuter instances

The provided morning commute data described users' origin and destination locations, and car capacity. Trips shorter than 5 kilometers or longer than 80 kilometers were rejected. An average driving speed of 60 kilometers per hour was considered, assuming that a large part of the commute passes highways. From the real-world data, ten instances were constructed by randomly selecting subsets of users. Figure 3 plots the coordinates of the users' origin and destination locations. The black dots indicate origin locations, while the red squares represent destination locations. Province borders are outlined in black. Pickup locations are clearly clustered in one main city while destinations are more geographically spread out.

¹The class A instances are publicly available at <https://people.cs.kuleuven.be/~pieter.smet/ridesharing>

Fig. 3: Geographic distribution of users' origin and destination locations in class B instances

5.1.3. Data analysis

Table 2 shows an overview of different characteristics for class A and B instances. The ratio of trip-to-time window length varies between both instance classes. For class A instances, this ratio is on average 0.60, while class B instances have tighter time windows with an average ratio of 0.78.

Figure 4 plots the distributions of trip lengths for both instance classes. Trip lengths in class A form a skew normal distribution while in class B three clusters of trip lengths can be identified. These three groups of trips correspond to the three groups of destination locations that are shown in Figure 3.

For all experiments, LAHC was given one hour of computation time. To account for its non-deterministic behavior, each run was repeated ten times with different seed values for the random number generator. Gurobi 8.0.0 was used to solve the integer programming problems, configured to use 16 threads. All experiments were conducted on a Dell Poweredge T620, 2x Intel Xeon E5-2670 with 128GB RAM.

5.2. Parameter tuning

5.2.1. LAHC list length

The first set of experiments analyzes the impact of the LAHC list length λ . Based on preliminary experiments, the following values for this parameter were selected for benchmarking: 1, 500, 1000, 5000, 10000, 15000, 20000 and 25000. Figure 5 shows, for each instance in classes A and B, the objective value for the aforementioned list lengths.

The results for class A instances may be categorized as follows. For instances A01 to A03, the value of λ has no significant impact on the final objective value. For these small instances, the same solutions were always obtained. Instances A04 to A09 confirm behavior previously observed with respect to LAHC on

Instance	Number of users				Trip length	Time window length	No. of pickup locations	No. of drop-off locations
	Total	Drivers	Flexible	Passengers				
A01	200	78	80	42	13.3	22.2	4.4	4.6
A02	400	151	167	82	13.3	22.1	6.6	7.0
A03	600	240	243	117	13.2	22.0	8.2	9.1
A04	800	319	338	143	13.4	22.4	11.5	12.0
A05	1000	409	392	199	13.3	22.2	15.4	16.0
A06	1200	487	472	241	13.2	22.0	18.5	18.3
A07	1400	555	567	278	13.3	22.1	19.8	20.3
A08	1600	652	644	304	13.3	22.2	24.8	24.3
A09	1800	741	712	347	13.2	22.1	26.2	26.5
A10	2000	818	780	402	13.1	21.9	27.5	27.4
A11	2200	884	869	447	13.2	22.0	31.7	31.9
A12	2400	942	963	495	13.3	22.2	33.6	32.9
A13	2600	1016	1069	515	13.2	22.0	36.4	36.1
A14	2800	1100	1156	544	13.3	22.2	39.4	40.0
A15	3000	1242	1187	571	13.3	22.2	43.4	44.1
B01	200	74	93	33	42.4	54.6	7.1	1.0
B02	400	174	157	69	45.5	58.5	12.6	1.0
B03	600	238	237	125	46.2	59.4	13.7	1.0
B04	800	315	334	151	45.5	58.5	20.0	1.0
B05	1000	399	398	203	45.4	58.3	25.1	1.0
B06	1200	479	491	230	44.3	57.0	29.5	1.0
B07	1400	547	576	277	44.9	57.8	34.7	1.0
B08	1600	629	660	311	45.2	58.1	41.4	1.0
B09	1800	704	738	358	45.0	57.9	46.9	1.0
B10	2000	793	819	388	45.0	57.9	51.9	1.0

Table 2: Overview of class A and B instances

other optimization problems: list lengths greater than one result in better solutions (Burke and Bykov, 2017). The best choice for λ ultimately varies depending on the given instance, but in general, larger instances favor shorter list lengths. For instances A10 to A15 and for all of the instances in class B, a list length of $\lambda = 1$ resulted in the best solutions. An analysis of the search progress for these instances revealed that the algorithm did not converge within the time limit. As an example, Figure 6 plots the objective value of the best solution S^* in function of the time elapsed for instance A10. For different values of λ , new best solutions are still found near the time limit, indicating that the algorithm has not yet converged.

5.2.2. Neighborhood selection probability

The probability of a specific neighborhood being selected was tuned using irace (López-Ibáñez et al., 2016). Let ρ_n be the probability with which neighborhood n is selected at line 6 in Algorithm 1. Values between 5% and 25% were considered for ρ_n , scaled such that $\sum_n \rho_n = 100\%$. Table 4 shows the values of ρ_n in the best parameter configurations found by irace.

In all tuned configurations, neighborhoods CO, AU and MU tend to be assigned high selection prob-

Fig. 4: Distribution of trip lengths for class A and B instances

Configuration	ρ_{CO}	ρ_{AU}	ρ_{MU}	ρ_{CL}	ρ_{DU}	ρ_{RU}	ρ_{SU}
1	20%	20%	14%	9%	14%	12%	10%
2	18%	21%	17%	6%	10%	7%	21%
3	24%	15%	13%	8%	11%	20%	8%

Table 3: Best parameter settings found by irace

abilities, while CL is always among the lowest-ranked neighborhoods. The remaining three neighborhoods - DU, RU and SU - receive varying selection probabilities which are generally lower than the first set of neighborhoods, but higher than CL. Note that these three groups of neighborhoods correspond to whether or not they directly manipulate the set of unassigned users Q . While this may indicate that some neighborhoods are more influential upon final solution quality than others, all neighborhoods contribute to the search process and are necessary to obtain the best possible solutions. Table 4 shows that the difference between the best parameter settings obtained by irace and when using an equal selection probability for all neighborhoods is small. These reported values are the gaps to the objective value obtained using equal selection probability for all neighborhoods. Positive values indicate that the corresponding configuration obtained better solutions. The best results are highlighted in bold.

5.3. Performance comparison

Table 5 compares the performance of LAHC with a general integer programming solver applied to the formulation presented in Appendix A. Both the time required by the solver to obtain the proven optimal solution (*Optimization time*) and the time required to generate all feasible patterns (*Preprocessing time*)

Fig. 5: Influence of LAHC list length λ on objective value

Fig. 6: Convergence plot for instance A10

are reported. For LAHC, the best solution obtained for each instance and the required computation time to find this solution are reported. The rightmost column shows the gap between the best LAHC solution and the integer programming solution.

All class A instances can be solved to optimality using the proposed integer programming formulation. The majority of time required by this approach is spent generating all feasible patterns, which can take up to 2.5 hours for the largest class A instance. LAHC finds solutions within 1% of optimality for instances A01-A07. The gap increases for larger instances, reaching up to 20% for the instance with 2800 users. The class B instances proved far more computationally intensive for the integer programming approach. The set of feasible patterns could be generated for instances B01-B03, while larger instances lead to out-of-memory exceptions. The optimization time required for these three instances is also comparatively higher than for the class A instances, while LAHC has a gap of around 9% to the optimal solutions.

5.4. Impact of problem properties

The results detailed in Table 5 demonstrate that the total number of users negatively impacts the solvers' performance. This section further explores which problem properties affect the performance of both LAHC and the proposed integer programming approach. In each experiment, one problem parameter is varied while the others are kept constant. The performance of each approach is quantified using different criteria. For the integer programming approach, the number of solved instances, the average number of generated patterns and the average computation time in seconds are shown. Meanwhile, for LAHC, the average gap to the optimal solution or lower bound is presented.

First, the impact of the number of flexible drivers is analyzed. For this experiment, all user are assumed to own a car, such that $|D| = |U|$, while varying the number of flexible drivers between 0% and 100%

Instance	Configuration 1	Configuration 2	Configuration 3
A01	0.00%	0.00%	0.00%
A02	0.00%	0.00%	-0.01%
A03	0.01%	0.00%	0.00%
A04	1.45%	0.29%	1.16%
A05	-0.39%	-0.97%	0.59%
A06	0.37%	0.92%	0.00%
A07	-0.31%	0.63%	0.32%
A08	-0.54%	-0.36%	-0.70%
A09	1.37%	1.21%	0.31%
A10	1.97%	1.37%	1.82%
A11	0.13%	0.54%	-2.00%
A12	0.93%	0.95%	0.77%
A13	1.06%	0.72%	0.35%
A14	2.10%	3.11%	0.84%
A15	3.86%	1.98%	0.99%
Average	0.80%	0.69%	0.30%
B01	-1.87%	-1.07%	-0.27%
B02	-0.20%	-0.15%	-0.62%
B03	-0.68%	-0.25%	-1.47%
B04	1.41%	1.51%	2.00%
B05	0.44%	-0.26%	0.03%
B06	0.39%	0.31%	0.18%
B07	0.38%	0.15%	0.53%
B08	0.12%	0.09%	-0.12%
B09	0.36%	0.47%	0.87%
B10	-0.04%	0.13%	-0.60%
Average	0.03%	0.09%	0.05%

Table 4: Gap in objective value of tuned configurations to equal selection probability for all neighborhoods

of the total number of drivers. Table 6 compares the performance of the two solution approaches for the instances from classes A and B using the aforementioned criteria. While all class A instances can still be solved within the time limit, the required computation time increases significantly when there are more flexible drivers. In general LAHC finds solutions close to the optimal solutions, however, the gap becomes larger when the number of flexible drivers increases. The instances from class B appear to be computationally harder for the integer programming approach. When more flexible drivers are considered, the number of patterns increases significantly and solutions can no longer be found within the time limit. The LAHC gaps are larger compared to those with respect to class A's instances, and also appear to increase with more flexible drivers. However, since optimal solutions are no longer found by the integer programming approach, these values now represent the gap to the lower bound obtained by the integer programming solver.

The second problem property which is analyzed is the maximum allowed detour. This property is defined relative to the driver's travel time and is varied from 5% to 45%. Table 7 again compares the two solution approaches on both instances classes. All class A instances were solved within the allowed time

Instance	Integer programming				LAHC		
	Optimization time (s)	Preprocessing time (s)	UB	Number of patterns	Best	Time (s)	Gap
A01	0.0	0.0	34015	214	34015	1.4	0.0%
A02	0.0	0.4	59023	449	59029	49.3	0.0%
A03	0.0	1.5	81003	782	81019	116.9	0.0%
A04	0.0	5.4	65279	1364	65316	822.0	0.1%
A05	0.0	14.2	97170	1688	97214	1537.8	0.0%
A06	0.0	33.3	100208	2248	101360	2323.6	1.1%
A07	0.0	85.5	119159	3026	120298	2803.4	0.9%
A08	0.0	237.8	99367	4549	107642	3112.5	7.7%
A09	0.1	342.3	116107	4929	126452	3172.0	8.2%
A10	0.1	572.2	111255	5539	125300	3190.5	11.2%
A11	0.1	1088.0	131191	6542	147264	3271.5	10.9%
A12	0.2	1995.3	102372	8448	124434	3202.1	17.7%
A13	0.1	3312.2	139254	9824	165269	2917.7	15.7%
A14	0.2	4997.0	107453	10851	135609	3137.4	20.8%
A15	0.2	8958.4	112433	14277	133698	2825.6	15.9%
<hr/>							
B01	0.1	2.7	5811	4615	6355	16.1	8.6%
B02	3.8	299.5	11449	35564	12485	175.3	8.3%
B03	120.9	641.6	19223	129182	21248	188.6	9.5%
B04	-	-	-	-	22029	269.4	-
B05	-	-	-	-	27275	429.4	-
B06	-	-	-	-	34823	418.3	-
B07	-	-	-	-	41983	434.6	-
B08	-	-	-	-	42725	446.4	-
B09	-	-	-	-	50032	446.5	-
B10	-	-	-	-	54191	448.7	-

Table 5: Comparison of integer programming and LAHC

limit, except for the largest instance when considering a maximum detour of 45%. As larger detours enable greater driver flexibility, the number of patterns increases and, consequently, the computation time required to generate them. LAHC only found optimal solutions when detours of up to 15% were allowed. For the other scenarios, the gap increased with higher flexibility. Again, class B instances proved significantly more difficult for the integer programming solver. Even when no detours were allowed, only four out of ten instances could be solved within the time limit. The reported LAHC gaps demonstrate that for the scenarios with small detours, near-optimal solutions are also obtained for the class B instances.

In a third series of experiments, the maximum walking time m_u was varied between 5 and 30 minutes. Table 8 compares the performance of the solution approaches. Given the structure of class A instances, increasing m_u to 20 minutes does not allow users to reach many alternative pickup or drop-off locations. While the number of patterns does not increase significantly between $m_u = 0$ and $m_u = 20$, there is a noticeable increase in required computation time. For $m_u \geq 25$, the integer programming approach could not find solutions within the time limit due to the large number of patterns which needed to be generated. LAHC demonstrates consistent behavior for these instances with an average optimality gap of 1.4%. As in the two previous experiments, the integer programming approach performed significantly worse on the class B instances. Increasing m_u did not negatively impact the performance of LAHC,

	Driver flexibility										
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Class A instances											
Number of instances solved MIP	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
Avg. number of patterns	1600.0	1792.8	1980.3	2169.7	2376.5	2572.1	2763.1	2960.7	3156.9	3341.6	3532.0
Avg. MIP computation time	0.0	8.6	26.1	49.7	82.0	133.6	193.0	262.8	373.9	472.8	619.1
Avg. LAHC gap	0.0%	0.3%	0.6%	0.9%	1.3%	1.6%	1.9%	2.2%	2.5%	2.8%	3.1%
Class B instances											
Number of instances solved MIP	10/10	10/10	7/10	6/10	5/10	4/10	4/10	4/10	3/10	3/10	3/10
Avg. number of patterns	1100.0	35583.7	-	-	-	-	-	-	-	-	-
Avg. MIP computation time	0.0	309.4	-	-	-	-	-	-	-	-	-
Avg. LAHC gap	0.0%	0.1%	0.7%	3.2%	6.9%	10.4%	12.0%	13.7%	13.9%	15.9%	16.3%

Table 6: Impact of driver flexibility on algorithm performance

	Maximum detour					
	0%	5%	15%	25%	35%	45%
Class A instances						
Number of instances solved MIP	15/15	15/15	15/15	15/15	15/15	14/15
Avg. number of patterns	1279.7	1283.4	1475.7	2476.9	5768.7	-
Avg. MIP computation time	94.0	112.5	126.5	357.8	1827.7	-
Avg. LAHC gap	0.0%	0.0%	0.0%	1.3%	10.0%	30.6%
Class B instances						
Number of instances solved MIP	4/10	2/10	1/10	0/10	0/10	0/10
Avg. number of patterns	-	-	-	-	-	-
Avg. MIP computation time	-	-	-	-	-	-
Avg. LAHC gap	3.5%	1.1%	1.0%	-	-	-

Table 7: Impact of maximum detour on algorithm performance

however, the gaps to the lower bound are higher compared to those reported in Table 7.

6. Simulations

6.1. The effect of driver flexibility

To demonstrate the effect of ride-sharing on overall solution cost, a series of simulations was set up in which the number of flexible drivers was varied between 0% and 100% of total drivers. In these simulations it was assumed that every user owns a car such that the subset of passengers $U \setminus D$ is empty and $|D| = |U|$. When there are no flexible drivers, each driver uses their own car to reach their destination and ride-sharing is not possible. The impact of having flexible drivers is quantified by the decrease in objective value relative to the baseline scenario in which there are no flexible drivers. Note

	Maximum walking duration						
	0	5	10	15	20	25	30
Class A instances							
Number of instances solved MIP	15/15	15/15	15/15	15/15	15/15	14/15	14/15
Avg. number of patterns	2424.5	2424.5	2424.5	2624.9	2624.9	-	-
Avg. MIP computation time	0.9	0.9	0.9	1638.8	1627.1	-	-
Avg. LAHC gap	1.3%	1.3%	1.3%	1.6%	1.6%	1.5%	1.5%
Class B instances							
Number of instances solved MIP	8/10	8/10	8/10	1/10	1/10	1/10	1/10
Avg. number of patterns	-	-	-	-	-	-	-
Avg. MIP computation time	-	-	-	-	-	-	-
Avg. LAHC gap	7.3%	7.3%	7.3%	4.8%	4.8%	4.7%	4.7%

Table 8: Impact of maximum walking time on algorithm performance

that the objective value in these simulations corresponds to the total distance traveled by drivers since users will never be unassigned. Figures 7 and 8 show the decrease in objective value for class A and B instances, respectively. A list length of $\lambda = 1$, equal neighborhood selection probabilities and time limit of one hour were used in all experiments.

Fig. 7: Impact of driver flexibility for class A instances

Ride sharing in an urban setting, as represented by the instances in class A, becomes up to 10% more efficient when drivers are willing to act as passenger. However, cost reduction suffers from diminishing returns when more than 30% of drivers are flexible. Nevertheless, these results provide a clear motivation for organizing ride-sharing, as even with half of the drivers being flexible, an average decrease of 7.5%

of the total costs may be realized. Note that, in general, the decrease is higher for larger instances, that is, the effect of organizing ride-sharing is more apparent with more participants. The difference in cost reduction between the smallest and largest instances is on average 3.9%.

Fig. 8: Impact of driver flexibility for class B instances

In general, the results obtained for the class B instances follow the same trends identified for the urban area instances: the realized cost reduction starts to converge at around 50% flexible drivers and higher reductions are obtained for instances with more participants. However, the actual decrease in objective value is much higher for these instances. With two-thirds of drivers being flexible, an average reduction of 36.1% is achieved, compared to 8.1% on the class A instances. This is due to the fact that the distances between users' origin and destination locations in class B instances are much larger compared to the class A instances. Reducing travel distance through ride-sharing has thus a much larger impact on the solution's objective value. The gap between the cost reduction obtained for the smallest and largest instances is 5.6%.

The results from these simulations lead to two main conclusions. Firstly, large cost reductions are realized when the number of flexible drivers is between 30% and 50% of available drivers. Having more than 50% flexible drivers results in marginal improvements. Secondly, the impact of having more flexible drivers is larger in scenarios with many participants and in scenarios in which the distances between users' origin and destination locations are large, as is the case in the commuter instances.

6.2. The effect of time windows

Previous experiments assumed that drivers were willing to extend their travel time by 25% to transport other users. To quantify the impact of having more driver flexibility in terms of time, a second series of simulations was performed in which the length of drivers' time windows was varied. The allowed amount of detour time was varied between 5% and 45% of the driver's travel time. As in Section 6.1,

the improvement in overall objective function was used as a metric, compared to the baseline scenario in which drivers are not willing to make any detour. Figures 9 and 10 plot the simulation results for class A and B instances, respectively. A list length of $\lambda = 1$, equal neighborhood selection probabilities and time limit of one hour was used in all experiments.

Fig. 9: Impact of maximum detour for class A instances

For all instances in class A, large reductions of the objective value were observed when larger detours were allowed. The biggest improvement was 90%, from 607,828 to 62,522, and was obtained for the instance A15 with 45% maximum additional detour time. The main cause of these huge improvements in objective value is that many more users can be assigned if drivers are allowed longer detours, which, due to the large weight associated with this objective term, has a strong effect on the overall objective value. Nevertheless, even with 25% detour flexibility, an average cost reduction of 33.5% is realized. In general, higher maximum detour values result in better solutions.

The results for the class B instances show a different trend compared to the results for class A instances. The improvements in objective value are much smaller compared to those obtained for class A instances. Up until 15%, the objective value decreases proportionally to the maximum allowed detour value. However, there is a clear stagnation of cost reduction for values higher than 15%. This phenomenon is again due to the penalties incurred by unassigned users: if more than 15% detour is allowed, all users can be assigned to a route. The small improvements at this stage are thus caused by decrease in total driving distance, which results in a much smaller decrease of objective value due to the assigned weights. This is in contrast to the class A instances in which, even with 45% allowed detour, there are still users left unassigned. Figure 11 plots, for different maximum detour values, the number of unassigned users in each instance. For class A instances, the number of unassigned users keeps decreasing for higher maximum allowed detours, while for class B instances all users are assigned when 15% or more detour flexibility is allowed.

In general, it can be concluded that increasing the drivers' maximum tour length is most beneficial in

Fig. 10: Impact of maximum detour for class B instances

Fig. 11: Number of unassigned users for class A and B instances

scenarios in which users' origins and destinations are geographically uniformly distributed. Scenarios in which locations are clustered do not require large detours of drivers to ensure all users get assigned to a route. It is worth noting that the main cost reductions came from assigning a larger number of users to one driver's route. The amount by which the actual distance driven was decreased was only marginal.

6.3. The effect of maximum walking time

The last parameter influencing user flexibility which is analyzed concerns the maximum time users are willing to walk to/from an alternative pickup/drop-off location. In a third series of simulations, the maximum walking time m_u was varied between 5 and 30 minutes. Assuming a speed of 4.8 kilometers per hour, alternative locations within 0.4 to 2.4 kilometers were added to the sets O_u^p and O_u^d . Figures 12 and 13 plot the obtained reductions relative to the baseline scenario with $m_u = 0$ for class A and B instances, respectively. Changes in both the number of unassigned users and total distance driven are shown. Preliminary experiments showed that increasing the number of alternative locations results in slower convergence of LAHC. Therefore, the experimental conditions were changed to a list length of $\lambda = 1$, equal neighborhood selection probabilities and a time limit of two hours, such that stronger convergence was obtained within the time limit.

Fig. 12: Impact of maximum detour for class A instances

No significant changes in solution cost were observed for the class A instances. Neither the total distance driven nor the number of unassigned users was significantly impacted by varying the maximum walking time. The number of alternative pickup and drop-off locations did increase with higher maximum walking time, but given the drivers' available car capacity and location distribution, this did not significantly affect solution quality.

For all instances in class B, the total distance traveled by the drivers increased for larger values of m_u , while the number of unassigned users did not always decrease (for example in B04). As a result, in some cases a worse solution was obtained with larger values of m_u . This is somewhat counter-intuitive. The objective value of an optimal solution can never be higher when users may consider more alternative locations as there are no costs associated with the use of alternative locations. The fact that such

Fig. 13: Impact of maximum detour for class B instances

solutions were produced indicates that LAHC was not able to reach good solutions within the time limit. Nevertheless, the reported changes in objective value are small, ranging between 1% and 3%.

7. Conclusions and future work

The present paper introduces a new decision support model for optimizing ride-sharing which integrates both assignment and sequencing decisions. The proposed model emphasizes participant flexibility, both in terms of which role they adopt and how willing they are to relocate to an alternative service location. Given the focus on solving large-scale problem instances, a late acceptance hill climbing metaheuristic is proposed which integrates several inter- and intra-route neighborhoods.

The computational study presented results from several simulations performed on two types of instances representing ride-sharing in an urban area and the morning commute from home to work locations. The outcomes of these simulations gave insights into the conditions under which ride-sharing results in cost reductions. It was demonstrated that the number of flexible drivers has a significant impact on solution cost, especially in scenarios which involve large distances such as the commuter instances. The simulations indicated that large cost reductions can already be obtained if only 30% of drivers are flexible. Having more flexible drivers still improved the overall solution but suffered from diminishing returns. In large cities, locations are uniformly distributed geographically, that is, origin and/or destination locations are not clustered. In such scenarios, increasing the detour length of drivers leads to more users getting assigned to a route. Overall, the impact on actual distance driven was limited. Finally, the

effect of considering more alternative pickup and drop-off locations was analyzed. For the considered instance classes, no significant impact was observed, however, results indicated that under these conditions it was harder for LAHC to converge to a local optimum.

The cost reductions discussed in the computational study have a strong environmental impact. If 50% of the available drivers are willing to also act as passenger, the total distance driven decreased by 34% in the commuter instances. This translates to a reduction of 1135.8kg of CO₂ emissions in a scenario with only 600 users when considering an average of emission of 123g CO₂ per kilometer driven (Transport and Environment, 2015). Note that if this optimization occurs twice per day, as is typically the case in carpooling, already a daily CO₂ emission reduction of two metric tons may be realized in a scenario with only a limited number of participants. Such results demonstrate the proposed model's role as a decision support tool for strategic decisions based on the quantified trade-off between reduction in CO₂ emissions and required participant flexibility.

The present paper analyzed the effect of different types of participant flexibility separately. Future research should focus on investigating and modeling the interaction between different degrees of freedom to better understand what drives the cost in ride-sharing systems. Research concerning dedicated algorithms and policies to address dynamic ride-sharing also presents a relevant contribution to the domain. Analyzing solutions for the dynamic problem may serve to validate the performance of different types of ride-sharing configurations designed based on the outcomes of the present contribution. Finally, the computational study indicated that the proposed integer programming formulation may benefit from applying a decomposition-based approach to solve large-scale problem instances to optimality.

Acknowledgments

The author would like to thank Greet Vanden Berghe for her advice and comments which improved the quality of this work. The author would also like to thank Jan Christiaens for his support in creating the figures and Luke Connolly for editorial consultation.

References

- Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* 223, 2, 295 – 303.
- Baldacci, R., Maniezzo, V., Mingozzi, A., 2004. An exact method for the car pooling problem based on lagrangean column generation. *Operations Research* 52, 3, 422–439.
- Bolaji, A.L., Bamigbola, A.F., Shola, P.B., 2018. Late acceptance hill climbing algorithm for solving patient admission scheduling problem. *Knowledge-Based Systems* 145, 197 – 206.
- Bruck, B.P., Incerti, V., Iori, M., Vignoli, M., 2017. Minimizing CO₂ emissions in a practical daily carpooling problem. *Computers & Operations Research* 81, Supplement C, 40 – 50.
- Burke, E.K., Bykov, Y., 2017. The late acceptance hill-climbing heuristic. *European Journal of Operational Research* 258, 1, 70 – 78.
- Chan, N.D., Shaheen, S.A., 2012. Ridesharing in north america: Past, present, and future. *Transport Reviews* 32, 1, 93–112.
- Chou, S.K., Jiau, M.K., Huang, S.C., 2016. Stochastic set-based particle swarm optimization based on local exploration for solving the carpool service problem. *IEEE Transactions on Cybernetics* 46, 8, 1771–1783.
- Cordeau, J.F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37, 6, 579 – 594.

- European Commission, 2015. Indicators on mobility and transport. Technical report, European Commission Joint Research Centre.
- Fonseca, G.H.G., Santos, H.G., Carrano, E.G., 2016. Late acceptance hill-climbing for high school timetabling. *Journal of Scheduling* 19, 4, 453–465.
- Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.E., Wang, X., Koenig, S., 2013. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological* 57, 28 – 46.
- Huang, C., Zhang, D., Si, Y.W., Leung, S.C.H., 2016. Tabu search for the real-world carpooling problem. *Journal of Combinatorial Optimization* 32, 2, 492–512.
- INRIX, 2017. Global traffic scorecard. <http://inrix.com/scorecard/>.
- Jørgensen, R.M., Larsen, J., Bergvinsdottir, K.B., 2007. Solving the dial-a-ride problem using genetic algorithms. *The Journal of the Operational Research Society* 58, 10, 1321–1331.
- Kaan, L., Olinick, E.V., 2013. The vanpool assignment problem: Optimization models and solution algorithms. *Computers & Industrial Engineering* 66, 1, 24 – 40.
- Lin, Y., Li, W., Qiu, F., Xu, H., 2012. Research on optimization of vehicle routing problem for ride-sharing taxi. *Procedia - Social and Behavioral Sciences* 43, 494 – 502.
- Long, J., Tan, W., Szeto, W., Li, Y., 2018. Ride-sharing with travel time uncertainty. *Transportation Research Part B: Methodological* 118, 143 – 171.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3, 43 – 58.
- Maniezzo, V., Carbonaro, A., Hildmann, H., 2004. An ants heuristic for the long-term car pooling problem. In *New Optimization Techniques in Engineering*. Springer, pp. 411–430.
- Parragh, S.N., de Sousa, J.P., Almada-Lobo, B., 2015. The dial-a-ride problem with split requests and profits. *Transportation Science* 49, 2, 311–334.
- Savelsbergh, M.W.P., 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing* 4, 2, 146–154.
- Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2015. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological* 82, Supplement C, 36 – 53.
- Stiglic, M., Agatz, N., Savelsbergh, M., Gradisar, M., 2016. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review* 91, Supplement C, 190 – 207.
- Transport and Environment, 2015. How clean are Europe’s cars? Technical report, European Federation for Transport and Environment.
- Van Den Dooren, D., Sys, T., Toffolo, T.A.M., Wauters, T., Vanden Berghe, G., 2017. Multi-machine energy-aware scheduling. *EURO Journal on Computational Optimization* 5, 1, 285–307.
- Wolfler Calvo, R., de Luigi, F., Haastrop, P., Maniezzo, V., 2004. A distributed geographic information system for the daily car pooling problem. *Computers & Operations Research* 31, 13, 2263 – 2278.
- Yan, S., Chen, C., Chang, S., 2014. A car pooling model and solution method with stochastic vehicle travel times. *IEEE Transactions on Intelligent Transportation Systems* 15, 1, 47–61.
- Yan, S., Chen, C.Y., 2011a. A model and a solution algorithm for the car pooling problem with pre-matching information. *Computers & Industrial Engineering* 61, 3, 512 – 524.
- Yan, S., Chen, C.Y., 2011b. An optimization model and a solution algorithm for the many-to-many car pooling problem. *Annals of Operations Research* 191, 1, 37–71.

Appendix A: Integer programming formulation

The considered ride-sharing problem may be formulated as an integer programming problem using a pattern-based formulation. For each driver $k \in D$, a set of patterns R_k is defined. A pattern is a feasible route for driver k in which users are picked up and dropped off at their destinations. The total number of patterns is impacted by the number of drivers, the maximum allowed detour and the maximum walking time. As shown in Section 5.4, when a large degree of flexibility is allowed, the number of patterns

becomes too large to enumerate.

The binary decision variable x_{kr} equals one if driver k is assigned to pattern r , and zero otherwise. A second binary decision variable y_u equals one if user u is part of a driver's route, and zero if user u remains unassigned. A binary parameter a_{ur} equals one if user u is part of pattern r . Let c_r be the total distance of pattern r . The problem may now be formulated as follows.

$$\min \sum_{k \in D} \sum_{r \in R_k} c_r x_{kr} + \sum_{u \in U} C y_u \quad (\text{A1})$$

$$\sum_{k \in D} \sum_{r \in R_k} a_{ur} x_{kr} + y_u = 1 \quad \forall u \in U \quad (\text{A2})$$

$$\sum_{r \in R_k} x_{kr} = 1 \quad \forall k \in \bar{D} \quad (\text{A3})$$

$$\sum_{r \in R_k} x_{kr} \leq 1 \quad \forall k \in \tilde{D} \quad (\text{A4})$$

$$x_{kr} \in \{0, 1\} \quad \forall k \in D, r \in R_k \quad (\text{A5})$$

$$y_u \in \{0, 1\} \quad \forall u \in U \quad (\text{A6})$$

The objective function (A1) minimizes the sum of the total distance driven and the number of unassigned users weighted with the fixed cost C . Constraints (A2) make sure each user is assigned to exactly one route or updates the corresponding decision variable y_u if a user remains unassigned. Constraints (A4) ensure exactly one pattern is assigned to every obliged driver. Constraints (A5) assign at most one pattern to every flexible driver, thereby allowing flexible drivers to also act as passengers. Finally, Constraints (A5)-(A6) enforce bounds on the decision variables.