INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

Intl. Trans. in Op. Res. 29 (2022) 1417–1453
DOI: 10.1111/itor.13058

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

The static on-demand bus routing problem: large neighborhood search for a dial-a-ride problem with bus station assignment

Lissa Melis* and Kenneth Sörensen

Department of Engineering management, University of Antwerp, Prinsstraat 13, Antwerp 2000, Belgium
E-mail: lissa.melis@uantwerpen.be [Melis]; kenneth.sorens@uantwerpen.com [Sörensen]

Received 13 October 2020; received in revised form 6 September 2021; accepted 6 September 2021

Abstract

We introduce a novel optimization problem to support the planning and routing of on-demand buses in an urban context. We call this problem *the on-demand bus routing problem* (ODBRP). Given are a fleet of buses with fixed capacity, a set of bus stations and travel times between them, and a set of transportation requests. Each transportation request consists of a set of potential departure and a set of potential arrival bus stations, as well as a time window, that is, an earliest departure time and a latest arrival time. The aim of the ODBRP is to (1) assign each passenger to a departure and arrival bus station and (2) develop a set of bus routes to fulfill each request in time while minimizing the total travel time of all users. We present the static version of the ODBRP, as well as a straightforward large neighborhood search heuristic to solve it. The performance of the heuristic is established by comparing it to an off-the-shelve heuristic solver (LocalSolver). We also use our heuristic to solve (slightly modified instances of) the well-known dial-a-ride problem. The results found by the heuristic for the on-demand bus system are compared to those of a simulated traditional public bus system with fixed lines and timetables. A thorough analysis of the comparison demonstrates that total user ride times can be significantly lower in an on-demand public bus system and shows that an on-demand bus system works best with a large number of small buses.

Keywords: transportation; public transport; metaheuristic; routing

1. Introduction

A traditional bus transportation system is composed of a set of lines that follow predefined routes and schedules. Passengers traveling from one station to another are required to plan their journey taking into account the fixed network and timetable. The inflexible nature of such a public bus system is not a good match for the poorly predictable travel patterns of a modern urban population (Nelson et al., 2010). Nowadays, cities are active 24/7 (Finn, 2012) and transportation outside

*Corresponding author.

normal working hours is common. Essentially, there is a mismatch between the flexibility of transport demand and the inflexibility of transport supply. Both during peak times and during off-peak times, the capacity offered by a traditional bus transport system does not match the ideal capacity. In 2013, the average occupancy rate of public transport (trams and buses) in Flanders (Belgium) was 24% (Vlaams Parlement, 2014). Such numbers, however, belie the fact that buses are filled to the brim during peak hours and run almost empty during the rest of the day. Some lines work at full capacity, while others are barely used. These inefficiencies cause extra costs and the loss of potential customers.

It is therefore not surprising that the popularity of demand responsive transport (DRT)—which includes taxi services (e.g., Uber, 2019a), but also buses and other vehicles that alter their route based on customer demand (such as the Turkish “dolmuş”)—has increased considerably over the last decades. Originally DRT was a service for niche markets, transporting the disabled and elderly, or active in regions with low demand. Such systems have grown to be a feeder system for passengers to reach the mass public transport network (Archetti et al., 2018; Alonso-González et al., 2018; Yu et al., 2021), motivating the introduction of the term *flexible transport services* (Mulley and Nelson, 2009; Nelson et al., 2010). Over the last few years several alternatives for collective public transport came into existence, mostly initiated by the private sector. Examples are car sharing (Katzev, 2003), bike sharing (Vergeylen et al., 2020), employee commuter programs, taxi and ride sharing (Agatz et al., 2012; Jung et al., 2016; Santos and Xavier, 2015), and on-demand carpooling. These alternatives, together with DRT systems, fit under the umbrella term *flexible urban transport* (FUT) (Finn, 2012). Research has found that FUT systems perform better when offered on a large scale (Archetti et al., 2018; Jokinen et al., 2011).

Notwithstanding the increase in FUT options, most public bus services in cities remain inflexible and based on fixed routes and time tables. This can—at least in part—be attributed to the fact that, until recently, it was impossible for a transportation supplier to obtain the necessary real-time information from its customers. For the first time in human history, however, mobile devices that can fulfill this function have become ubiquitous and automated vehicle location systems, which provide the real-time location of vehicles, are well established (Mageean and Nelson, 2003; Mulley and Nelson, 2009; Nelson et al., 2010). This recent development allows, at least in theory, a large-scale shift toward *on-demand public transport*, that is, a move from a system in which buses drive along fixed routes, to one in which buses drive along routes completely determined by passenger demand for transportation. In addition, the upcoming technology of autonomous buses would allow for substantial cost savings as driver costs vanish (Winter et al., 2018). As DRT systems commonly fail because of excessive costs, autonomous vehicles might tip the balance in favor of an on-demand system (Enoch et al., 2006).

In a fully on-demand public bus transportation system, a user would send a *transportation request* by indicating his departure location (usually the current location of the user) and his arrival location as well as his preferred arrival time (possibly “as early as possible”) using a mobile application. The system then responds by sending a proposal to the user telling him which bus station[1] he has to walk to and which bus will pick him up, as well as an estimated time of arrival. Upon confirmation, the booking in the system will be executed, delivering the passenger to his destination as efficiently as

[1]To avoid ambiguity, we will use the term *bus station* to refer to the physical location where a passenger can get off or on a bus, and reserve the term *bus stop* for the action of stopping the bus to pick up or drop off passengers.

possible (e.g., with as little possible waiting time). Because of the varying demand for transportation over time, bus routes and stops will be different every day and change throughout the day, making the system fully flexible. In this way, unnecessary empty kilometers are avoided and the quality of service for users improves. Routes are built from scratch based on incoming requests, there is no fixed sequence of stops that must be visited, and buses are not restricted to a geographical demand zone. These latter two characteristics distinguish a fully flexible system from the existing semiflexible systems (Errico et al., 2013).

In practice, passengers cannot be picked up and dropped off at their preferred locations. Particularly in urban environments, buses should stop only at clearly signposted bus stations that have been determined suitable and safe. An important decision in the design of an on-demand bus system is whether to allow the users to choose their departure and arrival bus stations, or whether to let the planning system handle this decision. We argue (and, later in this paper, demonstrate) that considerable efficiency gains are possible by letting the planning system *assign* bus stations to the departure and arrival of the users, of course within a reasonable walking distance of the passengers' actual departure and arrival locations. Even though the assigned stations might not be the ones closest to the (departure or arrival) location of the user, this decision allows the system to pool users in space and time, avoiding time-consuming unnecessary stops of the bus. Moreover, detours can be avoided if passengers are assigned in an intelligent way to stop along the route of a bus (e.g., just asking a passenger to cross a road and wait at a bus station in the driving direction of the bus can make a large difference). By assigning bus stations to passengers, the system can more efficiently plan all of the transport requests, to the benefit of all users. The set of possible bus stations can be fixed or time-dependent, when a station is available only during certain hours of the day.

The additional decision of bus station assignment (BSA) requires two decisions to be made simultaneously: (1) the assignment of arrival and departure bus stations, and (2) the routing of the buses. We have called the resulting optimization problem that combines both decisions the *on-demand bus routing problem* or ODBRP. To the best of our knowledge, this problem has not been described before in the literature.

In terms of problem structure, the ODBRP is a new optimization problem that combines elements of three existing problems: the *dial-a-ride problem* (DARP) (Cordeau and Laporte, 2007; Molenbruch et al., 2017), the *school bus routing problem* (SBRP) (Schittekat et al., 2013; Kim et al., 2012; Fügenschuh, 2009; Park and Kim, 2010; Ellegood et al., 2019) and *the pickup and delivery problem with time windows* (PDPTW) (Dumas et al., 1991; Ropke and Pisinger, 2006; Dahle et al., 2019). Table 1 shows the main features of these problems and highlights the ones that are shared with the ODBRP. In the ODBRP people are transported, but there is no maximum ride time constraint. The latter is contained in the single time window within which a passenger needs to be transported. In the DARP and PDPTW, each event (pickup or drop-off) has a separate time window and the SBRP has a time limit only on the arrival at the single drop-off location, the school. All three problems have constraints on the size of the vehicle, as does the ODBRP. The SBRP is the only problem that, like the ODBRP, does not have a maximum route duration. It is also the only problem that features bus stations to pick up and drop off passengers, as well as BSA. A single variant of the DARP using alternative nodes (a feature that can be compared to BSA) was found in Brevet et al. (2019). However, the alternative nodes are handled in a fundamentally different way: the algorithm first decides on the order of the requests, and calculates the best nodes using Dijkstra's algorithm afterwards. Unlike in the ODBRP, the decision on the alternative nodes and

Table 1
Features of the DARP, the SBRP, and the PDPTW, gray cells contain features these problems share with the ODBRP

DARP	SBRP	PDPTW
Transportation of people	Transportation of people	Transportation of goods
Max. ride time constraint	Max. ride time constraint	No max. ride time constraint
Separate time window for pickup and delivery	Delivery time window	Separate time window for pickup and delivery
Capacity constraints	Capacity constraints	Capacity constraints
Max. route duration	No max. route duration	Max. route duration
Multiple stations for departure and arrival	Multiple stations for departure, only 1 arrival station	Multiple stations for departure and arrival
Pickup/drop-off anywhere	Pickup at bus station	Pickup/drop-off anywhere
(No) bus station assignment	Bus station assignment	No bus station assignment

the routing does not happen simultaneously. In addition, only small instances of this problem are solved (maximum 371 requests), which does not correspond to an urban context.

In reality, the ODBRP will be highly dynamic, with requests being introduced in the system on a near-continuous basis. As is common, however, we first study the static variant of the problem and leave the dynamic variant for future research. In the *static ODBRP*, all requests for transportation are known in advance and routes are determined before the buses start their tours. Nevertheless, the static problem is not without its use cases. First, solutions of the static ODBRP can be used in a later stage to benchmark the performance of algorithms for the dynamic ODBRP. Second, the static ODBRP might be used, for example, by companies or groups of companies wishing to organize collective transport for their employees. In such a situation, it is reasonable to expect that users register their request for transportation some time in advance.

Summarizing, this paper makes the following *scientific contributions*: we introduce a novel optimization problem (Section 2), the ODBRP, which can be used for the planning and routing of on-demand buses in an urban environment, and includes BSA. We develop a first, straightforward heuristic to solve this novel optimization problem (Section 3). We derive some guidelines on the design of on-demand bus systems and demonstrate the significant benefits for the passengers (Section 4).

2. Problem description and formulation

In Section 2.1 a detailed description of the ODBRP is given using an example, as well as a description of an instance and a solution. In Section 2.2, the constraints of the problem are formally presented. A comprehensive mathematical model can be found in Appendix A.

2.1. Problem instance and solution

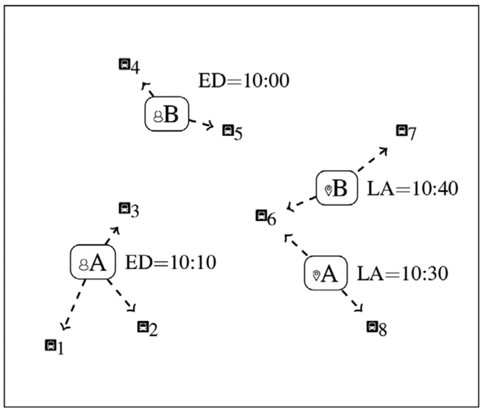
In the ODBRP, a set of bus stations, as well as a fleet of buses, is given. These buses can have identical or different capacities. A travel matrix, expressed in units of time, between each pair of bus stations is also given, as is a set of *transportation requests*. A transportation request is issued for a single passenger and consists of a set of (potential) origin/departure bus stations and a set of (potential) destination/arrival bus stations, along with a time window within which a passenger needs to be transported.[2] A passenger can only be picked up at one of his origin bus stations after his earliest departure time and has to arrive at one of his possible destination stations before his latest arrival time. The earliest departure time for a passenger at a certain station will differ depending on the walking time (WT) from the passenger's origin to this station, so that the departure time at the passenger's origin is the same, regardless of his assigned station. For example, when the assigned origin station is closer to the actual origin location of the passenger, the earliest departure time at this station will be earlier compared to when the passenger needs to do a longer walk to the chosen origin station. The same holds true for the latest arrival time, which is dependent on the WT of the passenger to his final destination, in such a way that the latest arrival time at the passenger's destination is the same regardless of the assigned arrival station.

Depending on the specific situation, three types of objective functions are possible for the ODBRP. The first type of objective function is profit related. When determining the routes of the buses we could minimize the travel costs (or total distance traveled), maximize the profit of the system (based on the revenues of requests), minimize the fleet size, minimize total driver wages, etc. A second type of objective function is completion related. Assuming it is mostly impossible to fulfill all the requests, the number of passengers served can be maximized. The last objective function type is quality related. In this category are objective functions that minimize total delay, total waiting time, total or maximum user ride time (URT), the number of passengers with delay, passenger travel time (PTT; including walking to a stop),... Delay is defined as the amount of time a passenger arrives at the destination station after his latest arrival time. Waiting time represents the time between the earliest departure time and the pickup time at the origin station. URT is the time between pickup and drop-off, when a passenger is actually on the bus, while travel time is the URT plus the waiting time and the WT to/from stations.

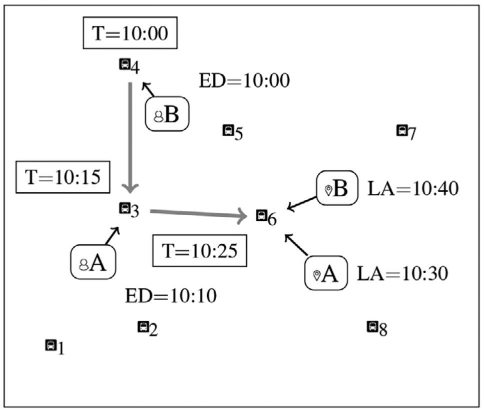
The choice of objective function will most likely hinge on the specific situation in which the ODBRP is solved, and may even change during the day. During off-peak hours, for example, when demand for transportation is lower, the fleet of buses is presumably large enough to serve all passengers. In this case it might be more interesting to consider profit- or quality-related objective functions. During rush hours, when demand is high, it is perhaps impossible to fulfill all requests for transportation and a completion-related objective function makes more sense. In the remainder of this paper, we will use a quality-related objective function, namely *the minimization of total user ride time (URT)*. This implies that all requests for transportation should be planned for a solution to be feasible. One might argue that the WT could be included in the objective function as this would optimize the passengers' entire journey. This is investigated in Section 3.3.

Figure 1a shows a visual representation of a small instance of the ODBRP with one bus, eight bus stations, and two users (A and B). Passenger A can walk to three bus stations for pickup and can be dropped off at two different bus stations. Also, each passenger has a time window. To simplify, we assume equal WTs to each of the possible stations for departure and arrival, making the time window independent of the assigned stations. In this case, passenger A can only be picked up after 10:10 a.m. and needs to arrive at a destination station before 10:30 a.m. Passenger B is able to walk

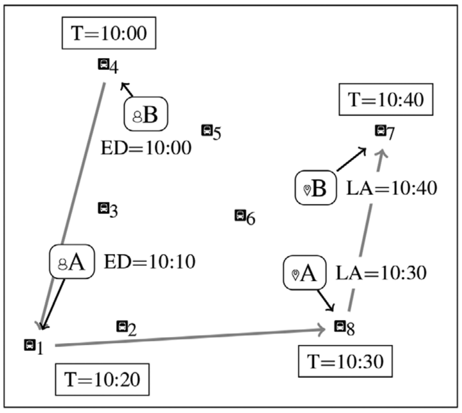
[2]We will use the terms origin and departure, as well as the terms destination and arrival interchangeably.



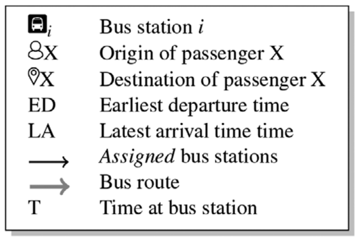
(a) Example instance



(b) Example solution *with* bus station assignment



(c) Example solution *without* bus station assignment



(d) Legend

Fig. 1. Example instance and solution: with and without bus station assignment.

to two bus stations for his pickup, which has to be after 10:00 a.m. For the arrival of passenger B, two stations are possible and the arrival has to happen before 10:40 a.m. A solution consists of two simultaneous decisions: (1) *Bus station assignment/selection*: assigning a departure and arrival bus station within walking distance to each passenger, and (2) *routing*: defining the sequence of bus stations visited by each bus.

In Fig. 1b, a feasible solution is represented for the instance in Fig. 1a. In this case, passenger A walks to the bus station in the northeast, takes the gray bus line and gets off the bus at the last stop. The time aspect is indicated in the figure in the boxes. Passenger A is picked up at his earliest departure time and is dropped off at 10:20 a.m., which is before his latest arrival time. BSA results in more flexibility for the routing as passenger pickup and drop-off can be grouped at bus stations to avoid extra stops during the route. In comparison, Fig. 1c visualizes a solution without BSA. In this case, passengers can choose their own bus stations for departure and arrival, for example, the stations closest to their actual departure and arrival locations. This results in less possibilities for efficient routing, and therefore longer bus routes.

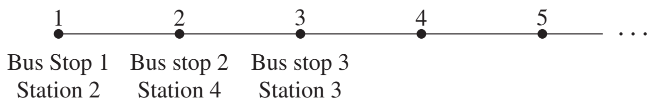


Fig. 2. A bus route in the ODBRP.

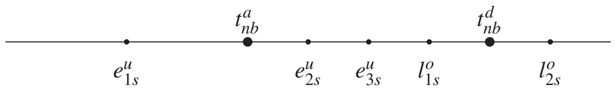


Fig. 3. Time aspect at bus stop.

2.2. *Formal problem formulation and constraints*

In this section, we present a formal description of the ODBRP with the minimization of the total URT as the objective function. The mathematical model can be found in Appendix A. The ODBRP can be formulated as follows.

In the ODBRP, a fleet B of buses is used to fulfill a set P of requests for transportation by visiting bus stations from the set of stations S. The objective is to minimize the total URT. Figure 2 shows an example of a bus route. Every bus $b \in B$ has a capacity C_b and the bus route of bus b consists of a number of filled positions $\{n_1, n_2, \dots, n_{|N|}\} \in N$. If position n_i is filled with a station $s \in S$, it is called a bus stop. This makes a bus route a consecutive sequence of bus stops of which the first stop is located at the first position. Each position in the solution can only be served once, which means that a bus can only stop at one station at the same time, but one bus station can be served multiple times (even by the same bus). A solution is then a set of $|B|$ different bus routes. In addition, at least one event (a pickup or a drop-off) is linked to each bus stop: a passenger getting on and/or off the bus.

A request from passenger $p \in P$ consists of a list of stations $S^u_p \in S$ within walking distance of the origin location, a set of stations $S^o_p \in S$ within walking distance of the arrival location of passenger p and a time window (an earliest departure time $e^u_{ps^u}$ and a latest arrival time $l^o_{ps^o}$ at the origin and destination station, s^u and s^o, respectively). The time window depends on the chosen origin or arrival station in such a way that the departure time at the passenger's origin location and the arrival at his destination location are the same regardless of the chosen station.

A solution of the ODBRP consists of (1) a set of assignments of passengers to departure and arrival stations, and (2) a set of routes visiting those stations. Bus stops have a time window linked by logical relations. If s is the nth stop of the bus and s' is the $n + 1$th stop of the bus, then the time of arrival at s' should be greater than the departure time at s plus the travel distance between stops s and s'. Also, the departure time (arrival time) of the bus at the nth stop is not smaller than the earliest departure time (latest arrival time) of a passenger assigned to that stop. Figure 3 shows an example of the time aspect at a bus stop. Three passengers are picked up and two passengers are dropped off at station s. The minimal latest arrival time of the passengers getting off the bus, in this case the latest arrival time of passenger 1, needs to be later than or equal to the arrival time at this bus stop. Similarly, the maximal earliest departure time of the passengers getting on the bus, needs to be earlier than or equal to the departure time of the bus at this stop. This allows a bus to wait at a bus station for extra passengers to get on the bus before leaving for the next stop.

A request p is served when a station from the set S^u_p is scheduled at an earlier position than a station from S^o_p in the same bus route, the event of this passenger getting on the bus at the stop from S^u_p happens after e^u_{ps} and the event of the passenger getting off the bus at the stop from S^o_p happens before l^o_{ps}. Of course a passenger is picked up before he can be dropped off, and both events need to be executed by the same bus.

Every bus has a capacity C_b. At no point can there be more than C_b passengers on the bus, while every request needs to be served to have a feasible solution. If a passenger is picked up at the nth position of bus b and dropped off at the n'th position of this bus, the URT of this passenger is defined by the arrival time of bus b at the n'th position minus the departure time of this bus at the nth stop. The sum of all the URTs of all passengers is minimized.

3. A heuristic for the ODBRP

The ODBRP is a problem closely related to the DARP, SBRP, and PDPTW, all of which are NP-hard. As a result, it is highly unlikely that an exact method will be able to solve instances of realistic size. A naive implementation of the mathematical model (found in Appendix A) in a commercial solver did not yield any useful results but for the smallest of instances. In this paper, we therefore focus on the development of an efficient yet straightforward *heuristic*, to be able to solve realistic instances of the ODBRP. In recent literature, large neighborhood search (LNS) heuristics are successfully used for the DARP, for example, Gschwind and Drexl (2019), Tellez et al. (2018), Masson et al. (2014), and Parragh and Schmid (2013), as well as for other ridesharing problems, for example, by Adawiyah et al. (2019). With these examples in mind, we propose a straightforward LNS heuristic. The aim of this heuristic is not to achieve maximum performance, something that is left for future research, but rather to present an intuitive and easy-to-understand solution method that may be used in our experiments (Section 4) to quantify the benefits of an on-demand public bus system.

Our method uses a greedy constructive heuristic both to generate an initial solution and to repair a partially destroyed solution. The constructive heuristic adds customers, one by one, in an intelligent and greedy manner. LNS iteratively destroys and repairs a solution until a termination criterion (either a time limit or a limit on the number of iterations) is met. Each iteration, after partly destroying and repairing the solution and if the solution is still feasible, local search (LS) is performed. We first explain the constructive heuristic and then discuss how it is embedded in the LNS framework. Appendix B goes into more detail on computational speedup measures.

3.1. *Constructive heuristic*

To build the initial solution, all passenger requests are first sorted on a *priority list* in ascending order of their latest arrival times. As we are not minimizing the number of buses used, all buses can operate. Starting from the top of the priority list, each request is allocated to one available bus,

Algorithm 1. Constructive heuristic: add one request p (Min. URT)

```
1  for Every bus b do
2      for Every position n in the bus do
3          Find origin station that causes smallest increase in route duration ;
4          Check feasibility (time windows and capacity violations) ;
5          if Feasible origin insertion then
6              for Every position $\geq$ n in bus b do
7                  Find arrival station that causes smallest increase in route duration ;
8                  Check feasibility (time window and capacity violations) ;
9                  Insertion criterion = $URT_p + \Delta_{URT} + Penalty$;
10                 if Feasible and insertion criterion < Best insertion criterion found then
11                     Save this insertion ;
12 if Feasible insertion found then
13     Perform best insertion ;
```

Table 2
Toy example—priority list

p	e^u_{ps}	l^o_{ps}	Departure stations	Arrival stations
1	40	100	1, 2, or 3	5 or 6
2	30	130	7, 8, or 9	10 or 11

until all buses have one allocated request. Then, requests are added one by one in a greedy manner with the constructive heuristic displayed in Algorithm 1.

Both for the origin and the destination of a request, every possible insertion position is checked in every bus route. To illustrate the constructive heuristic a toy example is used consisting of one available bus with capacity 8 and 2 requests. The priority list is shown in Table 2. For each request the following numbers are indicated: the passenger number, the earliest departure time, the latest arrival time, the possible stations for departure, and the possible stations for arrival. To simplify the toy example, the walking distances from the actual origin (arrival) location to each station out of the set of possible origin (arrival) station are assumed equal. As a consequence, the time windows become independent of the assigned station. The distance matrix between the stations is given in Table 3.

As mentioned above, the constructive heuristic first schedules one initial request in every available bus. Because the toy example has a fleet size of one bus, the first request is already planned in this bus. The outcome is visible in Fig. 4a. Bus stations 3 and 6 are chosen because they result in the smallest URT. The bus can only depart at time 40 because this is the earliest departure time of the first passenger. The distance between the two stops is equal to 10.

When adding the request of passenger 2 to the solution, we start by finding the first possible insertion position in the current solution to add the origin of passenger 2. In the example, there

Table 3
Toy example—distance matrix

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	0	5	5	20	15	13	20	25	20	35	35
2		0	5	20	15	14	15	15	20	35	35
3			0	20	15	10	12	14	14	25	30
4				0	30	35	40	40	40	60	60
5					0	5	30	28	32	40	41
6						0	30	28	32	40	38
7							0	5	5	40	45
8								0	5	45	50
9									0	45	50
10										0	5
11											0

are three possible origin insertion positions: before the first stop, in between stops 1 and 2, or after the last stop. We start by looking at the first: before the first stop. Because of BSA, the best origin stop is chosen from the list of possible origin stations of passenger 2, to enter in this insertion position. Because we insert before the existing route starts, there is only one existing neighboring station: station 3. The station causing the smallest increase in route duration is chosen. In the best case, station 3 would be present in the set of potential departure stations of passenger 2. This way, there would be no increase in route duration. However, this is not the case in this example. In this example, bus station number 7 is closest to 3 and therefore chosen as best origin station for this insertion. Figure 4b shows this possible origin insertion with an asterisk.

Every time a potential insertion location for the origin of a request is examined, the feasibility of the origin insertion is checked. The rationale for this is that, when this departure insertion is infeasible, it is pointless to look for an accompanying arrival insertion position. The feasibility check examines both the capacity and the time window violations of the possible insertion. The capacity check is trivial, the current capacity at every scheduled stop of every bus is being tracked. When an origin insertion is being checked, only the capacity at the departure stop is checked. If this stop is already at full capacity, continuing with this origin insertion is not possible. However, the full capacity check can only be done when trying to insert the arrival of a request, because the capacity of the bust might change (and therefore needs to be checked) between the departure insertion position and the arrival insertion position. Checking the time window constraints requires some additional calculations. When inserting a stop before, between or after existing bus stops, a detour occurs, which is calculated in minutes using the travel matrix. Once the detour has been calculated the algorithm examines whether it would cause any existing passengers in the route to arrive at their destination after their latest arrival time. If this is the case, the insertion is infeasible.

In the example, we calculate a detour of length 2, that is, the bus needs to drive two additional minutes when a new stop is inserted like in Fig. 4b. Because the latest arrival time of passenger 1 is 100, the insertion would be feasible. This means we can continue by looking for an arrival insertion position of which the position has to be larger than the one of the origin insertion position, as in the bus route drop-off has to occur after the pickup of a passenger. There are three possible insertion

Bus stop	1	2
Bus station	3	6
Arrival time	40	50
Departure time	40	50
P on	1	
P off		1

(a) Initial bus route

Bus stop	*	1	2
Bus station	7	3	6
Arrival time	30	42	52
Departure time	30	42	52
P on	2	1	
P off			1

(b) Possible origin insertion

Bus stop	*	*	1	2
Bus station	7	10	3	6
Arrival time	30	70	95	105
Departure time	30	70	95	105
P on	2		1	
P off		2		1

(c) Infeasible arrival insertion

Bus stop	*	1	2	*
Bus station	7	3	6	11
Arrival time	30	42	52	90
Departure time	30	42	52	90
P on	2	1		
P off			1	2

(d) Feasible arrival insertion

Bus stop	1	2	3	4
Bus station	3	6	8	10
Arrival time	40	50	78	123
Departure time	40	50	78	123
P on	1		2	
P off		1		2

(e) Consecutive placing of requests

Bus stop	*	1	2
Bus station	7	3	6
Arrival time	20	32	50
Departure time	20	40	50
P on	2	1	
P off			1

(f) Example origin insertion with waiting time ($e^u_{p_2 s} = 20$)

Bus stop	1	2	3	4
Bus station	7	3	6	11
Arrival time	20	32	50	88
Departure time	20	40	50	88
P on	2	1		
P off			1	2

(g) Example without time rescheduling after origin and arrival insertion passenger 2 ($e^u_{p_2 s} = 20$)

Bus stop	1	2	3	4
Bus station	7	3	6	11
Arrival time	28	32	42	80
Departure time	28	32	42	80
P on	2	1		
P off			1	2

(h) Example with time rescheduling after origin and arrival insertion passenger 2 ($e^u_{p_2 s} = 20$)

Fig. 4. Toy example.

positions for the arrival of this passenger: immediately following the departure position, between stops 1 and 2, or after stop 2.

For each possible insertion position, the algorithm looks for the best possible arrival stop for passenger 2. Figure 4c illustrates the first possible insertion of the arrival. The best arrival station for this insertion is station 10, because it extends the route the least. The rationale behind this is when choosing the best station for a certain insertion, minimizing the route extension automatically increases the URTs of already existing passengers in the route the least. Again, we have to do a feasibility check first. The capacity check is positive, but the time window check is not. This insertion would cause passenger 1 to arrive at time 105, after her latest arrival time, and is therefore infeasible. Figure 4d is an example of a feasible arrival insertion.

The previous paragraphs have described how the algorithm checks the feasibility of possible insertions of arrival and destination stops of a customer in a route. To decide which of the possible feasible insertions will be performed, an *insertion criterion* is used consisting of three parts. As inserting a new request might cause a detour or longer URT for the requests already planned in a route, the first part of the insertion criterion is the increase in total URT caused by this possible insertion. The second part is the URT of the passenger of the request that is going to be inserted. Thirdly, a penalty factor ensures that not all requests are planned consecutively. Figure 4e illustrates consecutive placing of requests. Passenger 1 is picked up and dropped off, then passenger 2 is picked up and dropped off. URTs are minimal, as the URT of each passenger is equal to the direct distance from a passengers' origin station to an arrival station in minutes. However, this would not be the most efficient when a large number of requests need to be inserted, possibly causing problems to plan every request, knowing that there is a limited bus fleet.

The penalty factor mentioned above contains only delays caused by inserting the origin bus stop of the passenger (e.g. the two-minute delay shown in Fig. 4b) or increases of the route length caused by the insertion of either the origin or the arrival bus stop of the passenger (an extension of the route of 38 minutes by inserting station 11, as illustrated in Fig. 4d) to avoid consecutive placing of stops. For the first block of requests, the penalty factor is multiplied by M. The magnitude of this block of requests is chosen to be five times the fleet size. M is a random number between 0 and 1 to ensure that the weight of the penalty factor varies and is not too large. This is important as the penalty factor is not actually optimizing the objective function, as it does not contribute to minimizing the total URT. After the first block of requests is inserted, M remains 0, that is, no more penalty factor is used. In the example shown in Fig. 4d, the insertion criterion would be $0 + 60 + (2 + 38)M$. The first part is 0 because the insertion does not cause the existing total URT in the route to rise. The second part is the penalty factor for making the route longer and the last part is the URT of the request that is to be added. Depending on M, the value of this criterion is between 60 and 100. In the example where requests are consecutively placed (see Fig. 4e), the insertion criterion equals $0 + 45 + (28 + 45)M$. The first part is 0 again, with the same reason as in the previous example. The second part is the penalty factor for extending the route and the last part is the URT of the newly added requests, namely the request of passenger 2. Depending on M, the insertion criterion is between 45 and 118. When one has to choose between the insertion implemented in Fig. 4d or the one in Fig. 4e, it would depend on M. If M would be 1, we would choose the first insertion. However, when this would be an insertion after the first block of requests, and M would be 0, we would choose consecutive placing of the requests.

When inserting stops either as the first or the last stop of a route under construction, the route length increases and some idle time might occur. Idle time occurs when the bus needs to wait for passengers to get on the bus, that is, when the arrival time of the bus at a certain stop occurs before the earliest departure time of a passenger who is assigned to get on the bus at this stop. For example, if the earliest departure time of passenger 2 would have been 20 ($e^u_{p_2s} = 20$) and assuming all other circumstances are the same, a waiting time of 8 would occur, as is illustrated in Fig. 4f. In the figure, we can see that this insertion would not cause the route to be delayed, thus idle time is not added in the penalty factor, however it is already incorporated in the insertion criterion as the increase in URT caused by inserting the passenger. Regarding the objective function, idle time is only disadvantageous when at least one passenger is on the bus during this time. The idle time is then called *waiting time*.

In the literature on DARPs a *time rescheduling procedure* is sometimes introduced (e.g., in Parragh et al., 2009). For a given route, such a procedure attempts to reschedule the departure and arrival times of the bus at the stops to maximally erase waiting times, and consequently decreases the total URT. In the example of Fig. 4f, the pickup of passenger 2 could be delayed to time 28, so that the waiting time disappears (for a comparison of solutions with and without a time rescheduling procedure, see Fig. 4g and h). However, in contrast to the cited work, the ODBRP is designed specifically to handle large-scale instances. In the case of the ODBRP, therefore, a time rescheduling procedure comes at a significant cost: it complicates the algorithm and increases the computation time significantly. Most importantly, however, in the instances used in this paper virtually all waiting times are erased by the algorithm without a time rescheduling procedure (see Section 3.3 for a brief analysis), leaving very little margin for such a procedure to have any effect on the objective function value. Because of the added complexity and related increase in computing time, combined with the limited benefits of a time rescheduling procedure, we have not included such a procedure in the algorithm.

After checking all combinations with the first possible origin insertion, all other possible origin insertions (and accompanying possible arrival insertions) in this and potentially other bus routes are also checked. The insertion with the smallest insertion criterion is performed. This is repeated for all requests.

3.2. Large neighborhood search

To further improve the solution, the constructive heuristic (Section 3.1) is embedded in an LNS framework, of which an overview can be found in Algorithm 2. The main idea underlying this framework (also called *destroy and repair* or *ruin and recreate*) is to iteratively destroy (part of) the solution and rebuild it using a destructive/constructive operator, further called a D&R combo.

The destructive operator clears one or more bus routes by removing all stops from it. The constructive (or rebuilding) operator adds the requests previously scheduled in these bus routes back into the solution with Algorithm 1. We adopt two D&R combos. One focuses primarily on minimizing the total URT (further called D&R(minURT)), which is the actual objective and another focuses on serving as much passengers as possible (the D&R(maxP)). The algorithm can switch between the two D&R combos. There are two ways to switch to D&R(maxP): (1) when the initial solution is infeasible, that is, when the number of served passengers (R) is smaller than the number of requests ($|P|$), line 15 in Algorithm 2, and (2) when the working solution is infeasible for five consecutive iterations, line 25 in Algorithm 2. There is only one shift from D&R(maxP) to D&R(minURT): when the solution has attained feasibility. When this point is reached, the objective function value has usually increased considerably. Therefore, an attempt to restore the total URT is executed by doing 20 iterations of LS (explained in Section 3.2.1) before going to the D&R(minURT), line 5 in Algorithm 2. This number of iterations is empirically chosen. After 20 iterations, the probability of finding a better URT using only LS decreases enormously. In vehicle routing literature, it is not uncommon to allow the solution to adopt higher objective function values along the search procedure to escape local minima, for example, threshold accepting algorithms such as the BATA algorithm of Tarantilis et al. (2004). In this case, it is primarily used to escape infeasibility.

Algorithm 2. LNS for the ODBRP

```
Function MaxP( iterations):
    while $R < |P|$ and iterations < 1000 do
        D&R(maxP);
        iterations++;
    if $R = |P|$ then
        while iterations < 1000 do
            Local search 20 iterations;
            iterations += 20;

Function Main():
    Build initial solution with constructive heuristic (Section 3.1, Algorithm 1);
    iterations = 0;
    count = 0;
    if $R \neq |P|$ then
        maxP(iterations);
    while iterations < 1000 do
        D&R(minURT);
        iterations++;
        if $R = |P|$ then
            Local search;
            count = 0 ;
        else
            if count = 5 then
                count = 0;
                maxP(iterations);
            else
                count++;
```

The differences in characteristics between the two operator combos are summarized in Table 4. The D&R(minURT) operator can be seen as the default combo, and for smaller instances or for instances with a relatively large fleet size, only this combo is used. For larger instances or instances with a small fleet size compared to the number of requests, the results improve significantly by changing operators when necessary. The algorithm continues until the limit of 1000 iterations is reached.

The bus routes to clear for the D&R(minURT) destructive operator are selected according to a criterion, randomly chosen among one of the following three criteria: largest total URT, smallest number of passengers, and largest URT to distance ratio. The second covers the bus route with the smallest number of passengers served. The third criterion is determined by calculating the ratio of

Table 4
Characteristics of the two D&R combos: minURT and maxP

Objective	D&R(minURT) Minimizes total URT	D&R(maxP) Maximizes $\|R\|$
Destroy:		
No. of buses to clear X	$X \sim U(1, 0.05 \times \|B\|)$	$X \sim U(1, 0.30 \times \|B\|)$
Criteria to clear bus routes	Largest total URT, smallest no. of passengers served and largest URT to distance ratio	Smallest no. of passengers served, random, maximum amount of waiting time
Rebuild:		
Insertion criterion	$\text{URT}_p + \Delta_{\text{URT}} + \text{Penalty}$	Absolute value of the route length increase

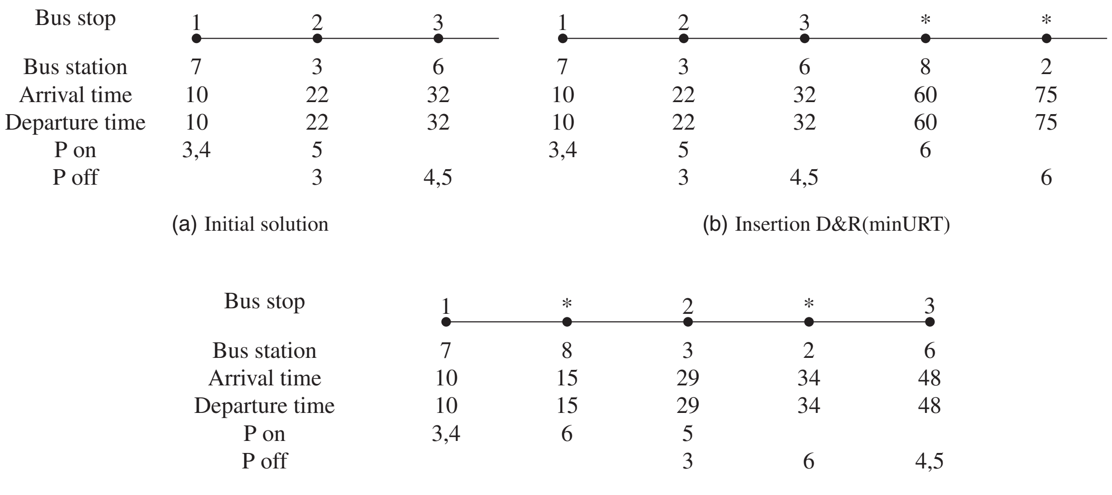
URT and direct distance in minutes from departure to arrival station for each passenger and summing the results per route. The number of bus routes to clear is decided as a uniformly distributed random number between 1% and 5% of the total fleet size. Because of the computation time needed to rebuild bus routes and the fact that we want to keep the most efficient bus routes in the solution intact, the number of bus routes to clear is deliberately kept low for the default operators.

When a criterion is chosen more than once, the routes that are next on the list according to this criterion are also cleared. For example, if the largest total URT criterion is chosen twice, then the bus route with the largest and the second largest total URT will be selected. The requests that were scheduled in the cleared bus routes are added to the list of unserved passenger requests.

In the next step, the emptied bus routes are first filled with the origin and destination stations of the first requests on the list of unserved requests. This list is not sorted (as was done when constructing the initial solution), rather the order of requests on this list depends on the (randomly chosen) order of the bus routes chosen for deletion. The passengers that were scheduled in the first bus route chosen to be cleared, come first in the priority list. Always adding the first passengers of the list to different empty vehicles, ensures the resulting solution is different, because these passengers were most likely scheduled in the same bus in the previous iteration. Finally, we try to add the other unserved requests to the current solution with the constructive heuristic explained in Section 3.1.

As was mentioned before, the change of operators happens when the algorithm is not able to serve every request, thus when the algorithm is working with an infeasible solution. We conduct the change immediately if the initial solution does not serve every request or after five consecutive LNS iterations of not serving everyone. In this case, it is desirable to have a feasible solution as quickly as possible. Therefore, the number of bus routes to clear is set to a higher number determined as a uniformly distributed random number between 1% and 30% of the total fleet size. The bus routes to clear are again chosen according to a criterion, randomly chosen among three criteria: the smallest number of passengers, random, maximum amount of waiting time. The constructive heuristic accompanying these destructive operators is very similar to the one explained in Section 3.1, it uses only a different insertion criterion.

A second toy example is shown in Fig. 5a, using the distance matrix shown in Table 3. Three requests are already planned in the bus route. At the first stop, passengers 3 and 4 are getting on the bus, at the second stop passenger 5. Passenger 3 gets off the bus at the second stop and the others



Bus stop	1	2	3
Bus station	7	3	6
Arrival time	10	22	32
Departure time	10	22	32
P on	3,4	5	
P off		3	4,5

(a) Initial solution

Bus stop	1	2	3	*	*
Bus station	7	3	6	8	2
Arrival time	10	22	32	60	75
Departure time	10	22	32	60	75
P on	3,4	5		6	
P off		3	4,5		6

(b) Insertion D&R(minURT)

Bus stop	1	*	2	*	3
Bus station	7	8	3	2	6
Arrival time	10	15	29	34	48
Departure time	10	15	29	34	48
P on	3,4	6	5		
P off			3	6	4,5

(c) Insertion D&R(maxP)

Fig. 5. Second toy example to illustrate difference between the different destroy and repair operator combos.

get off at the third stop. Also, the arrival and departure time of the bus at the stops is indicated. At this point, passenger 6 needs to be inserted in the schedule. In Fig. 5b, the best insertion when minimizing the total URT is demonstrated. The origin and destination stop of passenger 6 is added to the back of the route. Thus, the URT of the already existing passengers in the route is not changed. The total URT increases only with the URT of passenger 6, which amounts 15. The insertion criterion for minimizing the total URT thus equals 15. When we would be maximizing the amount of passengers served, the insertion criterion is equal to the absolute value of the route length increase. This criterion is also minimized because the smaller the route length increase, the more time is left to serve other passengers. In this case, the extension aggregates to 43, 22 for adding the origin stop and 15 for adding the arrival stop. In Fig. 5c, a different insertion is demonstrated. In this case, the insertion criterion for maximizing the number of passengers served is 7 for the origin stop (the arrival time of stop 2 now minus the arrival time of stop 2 in the initial solution) and 9 (same for stop 3) for adding the arrival stop of passenger 6, together this equals 16. As 16 is less than 43, this insertion is better when maximizing the number of passengers served compared to the one in Fig. 5b. When minimizing the total URT, the insertion criterion for the origin would become seven times the number of passengers affected, thus seven times 2, because both passengers 3 and 4 will spend more time on the bus as a result of this insertion. For the arrival, it would be nine times 2 because both passengers 4 and 5 are affected now. Also, the URT of passenger 6 has to be added to the sum. In this case, his URT would be 19 ($= 34 - 15$). The insertion criterion sums up to 51 ($= 7 \times 2 + 9 \times 2 + 19$). This is larger than the 15 from Fig. 5b. The toy example demonstrates the difference in final insertion choices between the two operators.

3.2.1. Local search

In addition, to the LNS framework, LS is included. LS makes small changes to the solution to improve the solution quality. In this case, we use a remove–insert operator on the entire neighborhood of the solution. Every request is randomly removed in turn, and if a better insertion position in terms of minimizing total URT is found in the same or a different bus route, the request is inserted there. If no better insertion can be found, the original insertion position of the request is kept. LS is used in every iteration if all requests are served. Also, LS is used to quickly restore the total URT after successfully maximizing the number of requests served until every request is served again. Contrary to destroying and repairing the solution, with LS it is certain that all requests remain served.

The LS component has a major influence on the solution quality, and is especially important when the algorithm has difficulties scheduling every request. This is further discussed in Section 3.3.

3.3. Algorithm and problem structure analysis

In this section algorithm design decisions on the D&R change, LS, the penalty factor, and exclusion of a time rescheduling procedure, are numerically substantiated. We also demonstrate the positive effects of BSA on the user's experience and discuss the decision to exclude the WT from the objective function.

In Appendix C, an algorithm quality assessment can be found. The proposed algorithm is compared to outcomes from the heuristic solver Localsolver (LocalSolver, 2020a), and results of benchmark DARP instances, converted to fit the ODBRP, are also presented.

In this section, instances are used with 121 equally distributed stations in a 100×100 Euclidean plane. The stations form a grid on the plane (the same grid of stations is used in Section 4 in Fig. 11). The driving distance between two adjacent stations is 10 minutes. A maximum of 2000 requests are randomly generated, of which the earliest departure times are all in the first hour of the time horizon. We assume that there is a homogeneous fleet of buses, with identical capacity.

3.3.1. LNS-operator change

As was explained before, *the D&R-change* is located from lines 1 to 7 in Algorithm 2 and is carried out when the algorithm has trouble finding a solution that serves every request. To study the benefit of the D&R-change, an example instance with 2000 requests and a fleet size of 250 buses with capacity 8 is used. The initial solution does not serve all requests (it serves only 1784 out of the 2000 requests) and a LNS-operator change is necessary to obtain a feasible solution. Figure 6a plots the number of passengers served as a function of time with and without using the operator change. Without it, no feasible solution is found, while with the D&R-change a first feasible solution is found after 1.6 seconds. The solution becomes infeasible twice during a computation time of two minutes, but is quickly restored each time. After these restorations the total URT is further improved.

3.3.2. Penalty factor

Figure 6b shows the contribution of *the penalty factor* explained in Section 3.1. Using the penalty factor, the algorithm finds the first feasible solution in 1.6 seconds, while this time almost doubles to 3 seconds without it.

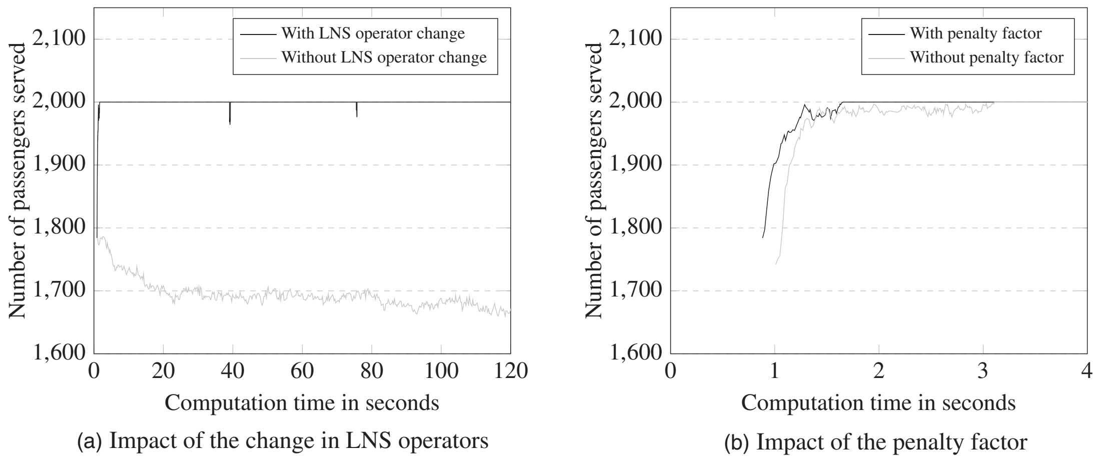


Fig. 6. The number of passengers served plotted for an instance with 121 stations, 2000 requests, 250 buses with capacity 8.

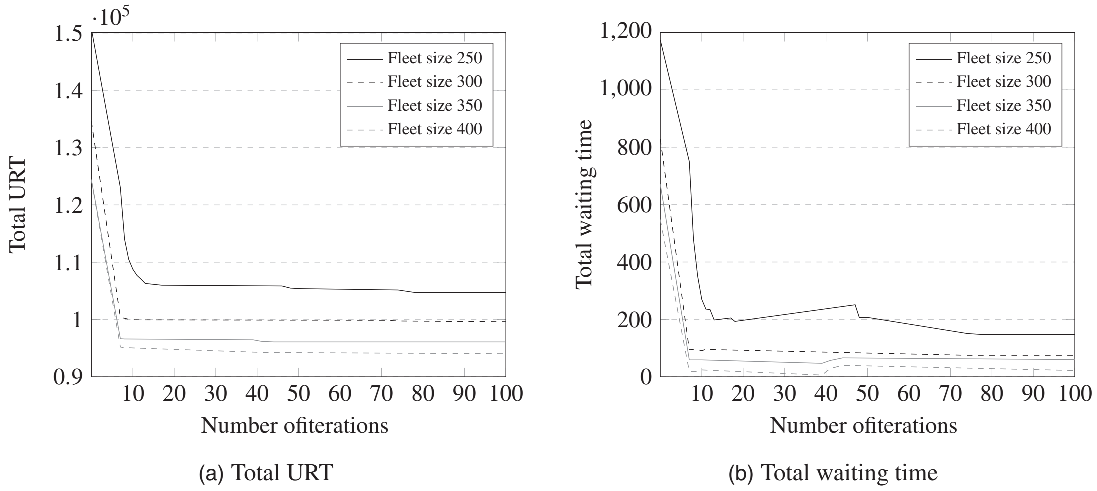


Fig. 7. Total URT and total waiting time for different fleet sizes, plotted for an instance with 121 stations, 2000 requests, buses with capacity 8.

3.3.3. Waiting time

We also analyze the occurrence of waiting time throughout the algorithm to support the design decision to exclude a time rescheduling procedure, as explained in Section 3.1. Waiting time occurs when passengers are on a bus that is standing still to wait for other passengers to get on. Figure 7 clearly shows that the total waiting time is minimized throughout the algorithm as part of the minimization of the total URT. Even though the waiting time is not entirely erased, doing so would cause only a 0.02–0.1% decrease in total URT, depending on the available fleet size, for the instance with 2000 requests. For smaller scale instances, the algorithm will often find a zero waiting time solution.

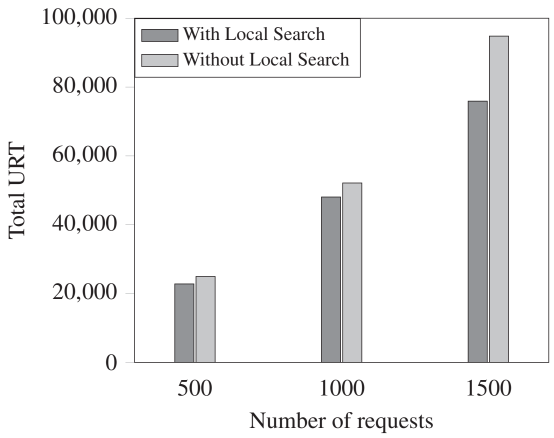


Fig. 8. Effect of local search (121 stations, 250 buses with capacity 8).

3.3.4. Local search

The influence of *the LS* component is investigated by running the algorithm with and without the LS component (i.e., lines 6 and 18 in Algorithm 2). Of course the LS is very time-consuming because every request is removed and reinserted into the solution. To make the comparison fair, the algorithm is run with LS for 1000 iterations and the average computation time needed for this configuration is used as a termination criterion to run the algorithm without LS. The fleet size is once again set to 250 and the capacity to 8. The differences in URT when using LS or not are relatively small (around 8%) for instances with, respectively, 500 and 1000 requests. This is shown in Fig. 8. However, when the number of requests increases compared to the fleet size, a larger difference can be seen: around 20% in total URT for instances with 1500 requests. Furthermore it should be noted that the last improvement in URT is found a lot faster when using LS. This indicates that adding LS results in significantly lower total URTs and is especially important when the fleet size is small compared to the number of requests.

3.3.5. Bus station assignment

Next, we briefly compare the effect of *BSA* on the total URT. When BSA is excluded, all passengers are allocated to their closest stop both for arrival and departure. Hence, passengers no longer have a set of potential departure and arrival stations, but the algorithm has to work with one station for the arrival and one for the departure. In an experiment, our algorithm was run both with and without BSA for two minutes. The results of these experiments are shown in Fig. 9. It is clear that the lack of flexibility significantly increases (i.e., worsens) the objective function value of the solutions produced by the algorithm. The larger the number of requests, the larger the difference in URT between the solution with and without BSA. For instances with 500 requests the difference is around 18% on average, while for instances with 1500 requests, it increases to 24%. Instances with more than 1500 requests are not incorporated in the graph because it became impossible to schedule all the requests when BSA was excluded.

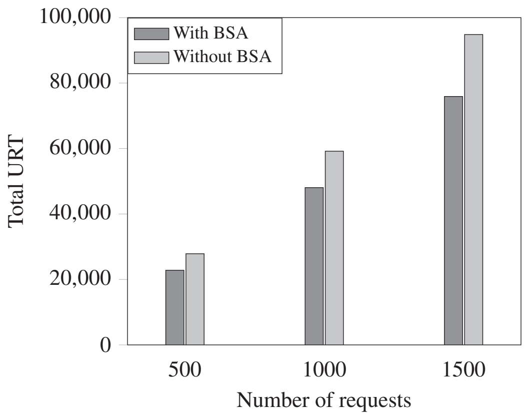


Fig. 9. Effect of bus station assignment (BSA) (121 stations, 250 buses with capacity 8).

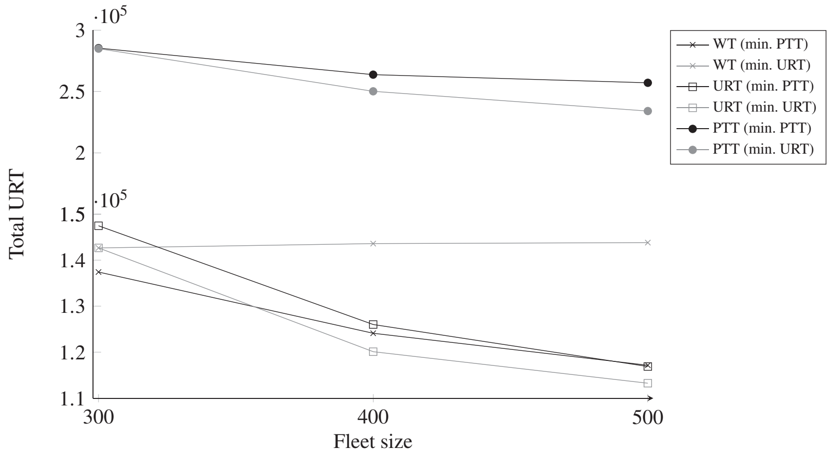


Fig. 10. Walking time analysis: minimizing total URT versus total PTT, for different fleet sizes (121 bus stations), 2000 requests, and buses with capacity 8.

3.3.6. Walking time

Finally, the inclusion of the *WT* in the objective function is analyzed. As mentioned in Section 2, in this work we minimize the total URT, which is the total time passengers spend on the buses. When the WT is included in the objective, *the total PTT* is minimized. For this analysis, we assume that walking is six times slower than riding the bus, which corresponds to an average walking speed of 5 km/h and an average driving speed of 30 km/h. Algorithmically, only a few things change. First, the insertion criterion for the D&R(minURT), which previously consisted of the increase in URT for the already scheduled passengers in the route, the URT of the to-be-added passenger and the penalty factor, now also includes the WT of the to-be-added passenger. The WT is dependent on the assigned bus stations for departure and arrival. Second, one of the criteria for clearing bus routes is the LNS changes. Instead of clearing bus routes with the largest URT, bus routes are cleared with the largest PTT. Figure 10 shows the differences in URT, WT, and PTT when minimizing the total

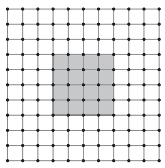


Fig. 11. Traditional public transport grid.

URT or PTT. Instances of 2000 requests are solved for different fleet sizes. Note that the scale on the *y*-axis is not evenly distributed to show all plots in a clear way. Logically when minimizing the PTT, total WT is lower compared to when the total URT is minimized. Moreover, the difference grows when the fleet size is larger. In this case, there is more flexibility to route the buses with small detours to ensure that passengers do not have to walk further than necessary. On the other hand, the total URT is smaller when minimizing the total URT compared to minimizing the total PTT. However, this difference seems almost independent of the fleet size. As a result, the total PTT is only lower when minimizing the total PTT *and* when the fleet size is large compared to the number of requests, that is, when the system behaves more like a taxi service than a public bus service. In the remaining parts of this work, the adopted fleet size is rather small. Therefore we continue to minimize the total URT.

4. Comparison to traditional public bus transport

In this section we compare the on-demand bus system (as modeled by the ODBRP) with a traditional bus system with fixed lines and fixed timetables. To this end, we model the traditional urban public bus transport network as a grid with 11 bus lines going north to south, and 11 bus lines going west to east, both back and forth (see Fig. 11). The colored square in the middle represents the city center, which we will call the inner city. The noncolored area is called the outer city. A bus station is located at every intersection of two lines, which results in a total of 121 bus stations. The standard frequency is set at one bus every 30 minutes and two adjacent stops are located at a distance of 10 minutes from each other.

To compute the total URT in the traditional public bus transport (TPBT) network, we use the same instances as for the ODBRP. Every request consists of one passenger, with a number of possible departure and arrival stations and a time window. The maximum walking distance between a passenger's origin (arrival) location and origin (arrival) station is set to 10. This means that the maximum number of possible departure or arrival stations is 4: the corner points of the square in the grid in which the passenger's arrival or destination is generated. Because buses run either horizontally or vertically, passengers whose departure and arrival cell are not in the same or adjacent rows or columns, will need to transfer. To simplify the solution of the traditional bus system, passengers are allowed to make only one transfer. If a passenger, for example, wishes to travel from the upper right corner to the lower left corner of the grid, there are two travel options: first north to south, then east to west, or the other way around. Given the fact that every passenger can choose between at most four stops, both for departure and arrival, this means that every passenger using the traditional network can have a maximum of 32 ($4 \times 4 \times 2$) ways to travel from his origin to his destination. For every request, the travel option with the shortest total URT is chosen.

Waiting time at home is not included in the total URT, as it is also not included in the URT calculated in the ODBRP. However, waiting time at a transfer station is included in the URT. Both in the ODBRP and TPBT, passengers need to be transported within their given time window. Not including the waiting time at the first stop is not entirely realistic. However, we want to give some advantage to the TPBT to ensure that the ODBRP actually performs better and that the improvement is not caused by the way of comparing the two systems.

After calculating the best total URT of the instance with the TPBT network, we know which lines are used, and the number of buses needed to serve those lines is partly optimized. This means that if a line (defined by a certain departure station and time) was not used in the TPBT, no bus is allocated to this line. We assume equal costs for both systems. Especially, with the rise of autonomous electrical vehicles as was mentioned in Section 1, the cost per kilometer declines as the driver's cost vanishes, making this a reasonable assumption. Subsequently, the ODBRP-heuristic explained in Section 3 is used to approach the total URT in the on-demand system. The same requests and bus stations are used, and the fleet size is set to the number of buses actually used in the traditional system.

The LNS heuristic is run for 1000 iterations for every instance. Six instances are presented, each with a different demand pattern: randomly distributed demand (U, the benchmark instance), triangle distributed demand with the actual city center (node (50,50)) as the mode (T), only inner city demand (II), only outer city demand (OO), all requests from outer to inner city (IO), and all requests from outer to inner city (OI). The instances also differ in number of requests (500–2000), but the 500 requests in a 500-request-instance are present as the first 500 request in a 1000-request-instance of the same type, and so on. The earliest departure times of the requests are all in the first hour of the time horizon. The latest arrival times are set according to the distance that needs to be traveled. In this case, we have set the latest arrival time equal to the earliest departure time plus two times the (Euclidean) distance that needs to be traveled (in minutes) plus 20 minutes. The fixed term of 20 minutes is chosen to ensure that requests with short distances still have a reasonable time window. When WT to the assigned stations is shorter than the maximum of two times 10 minutes (departure and arrival), the time window is adapted accordingly.

To start, the frequency of the fixed lines is set to 30 and the capacity constraint is deleted from the problem. This way, fixed public transport is not disadvantaged as it generally uses large-sized vehicles, The results are presented in Table 5. Next to the instance size, including the fleet size necessary to operate the TPBT with frequency 30 and the total URT for both systems, also the URT per direct distance for the ODBRP is given. The average, quartiles, and median give an idea of the overall quality of the solution, while the maximum URT per direct distance provides an indication of the passenger who is served in the most unfortunate way. When demand is randomly distributed, which is representative for a city with diverse points of interest, the on-demand system performs 23.5% better than the TPBT when scheduling 500 requests. This percentage declines to 10% when the number of requests is 2000. The latter can be explained by the increase in URT for the ODBRP when the number of requests is relatively high compared to the fleet size. Logically, an on-demand system will never replace, for example, a high-capacity metro line. This finding is corroborated by the results for the inner city instance (II). In this instance, all demand is located in the city center. The fleet size necessary to serve all requests in the TPBT system is more than halved, resulting in an

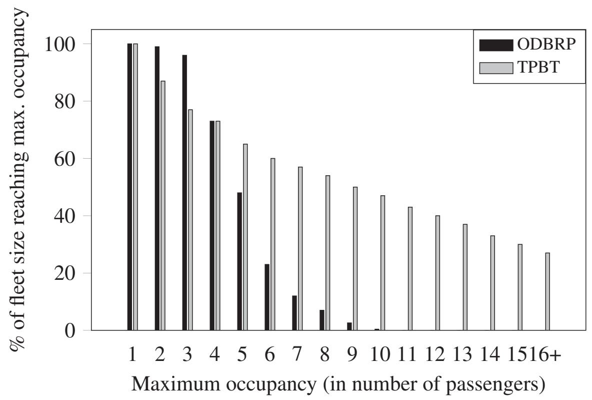
Table 5
Results: frequency fixed lines 30 and no capacity constraints

	Size		URT			URT ODBRP/direct distance				
Instance	*n*	*b*	TPBT	ODBRP	% Δ	max	Average	Q1	Median	Q2
U	500	188	30,860	23,599	−23.5	1	1	1	1	1
T	500	142	20,580	15,757	−23.4	1	1	1	1	1
II	500	70	10,490	85,561	−18.3	1.34	1.01	1	1	1
IO	500	135	23,610	17,754	−24.8	1.26	1	1	1	
OI	500	134	22,800	17,477	−23.3	1.30	1.01	1	1	1
OO	500	192	33,810	25,658	−24.1	1.02	1	1	1	1
U	1000	210	60,650	47,341	−21.9	1.43	1.01	1	1	1
T	1000	158	40,250	32,647	−18.9	1.50	1	1	1	1
II	1000	74	21,130	/	/	/	/	/	/	/
IO	1000	159	47,300	38,512	−18.6	1.65	1.05	1	1	1.07
OI	1000	148	546,320	412,251	−11.0	1.98	1.09	1	1.03	1.13
OO	1000	208	65,640	51,181	−22.0	1.45	1.01	1	1	1
U	1500	218	99,102	76,107	−16.4	2.10	1.05	1	1	1.07
T	1500	165	60,790	56,531	−7.0	2.56	1.11	1	1.02	1.18
II	1500	77	31,560	/	/	/	/	/	/	/
IO	1500	175	70,930	64,913	−8.4	3.04	1.12	1	1.06	1.17
OI	1500	165	69,190	70,027	+1.2	2.96	1.20	1	1.11	1.30
OO	1500	222	98,600	81,625	−17.2	1.74	1.04	1	1.01	1.06
U	2000	226	120,910	108,813	−10.0	2.13	1.11	1	1.06	1.17
T	2000	172	80,520	/	/	/	/	/	/	/
II	2000	80	42,500	/	/	/	/	/	/	/
IO	2000	179	94,140	/	/	/	/	/	/	/
OI	2000	178	93,080	/	/	/	/	/	/	/
OO	2000	226	131,570	116,415	−11.5	2.51	1.09	1	1.04	1.14

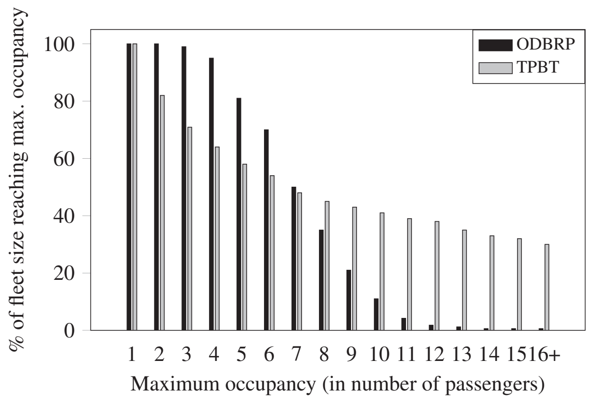
impossibility of the on-demand system to serve all requests, when the number of requests is larger than 500.

In general, the larger the number of requests, the more the differences between instances become apparent. When the number of requests is 1500, all instances, except the inner city one, can still serve all requests. However, percentage differences between the total URT of the on-demand system and the TPBT system are starting to drop. Especially, when all requests are coming from the outer to the inner city, the advantages of an on-demand system are completely diminished, compared to a 16.4% difference for randomly generated demand. This can be explained by the smaller fleet size necessary to operate the TPBT, but especially by the fact that all passengers want to depart from very distinct locations within the same hour. This is the major difference with the IO instance, where all passengers want to leave the smaller sized inner city within the first hour of the time horizon. The latter is easier to schedule, and results in a 8.4% difference in favor of the on-demand system.

The statistics on the total URT per direct distance show that the majority of the requests have a URT of decent quality, with a maximum of 1.30 as the second quartile. The direct distance used to calculate these figures, is the one from the assigned station for departure to the assigned station of arrival, thus only the direct distance that needs to be traveled by bus. In the time windows, also WT is included, therefore the maximum URT per direct distance can increase to a maximum of 3.04.



(a) U-instance, fleet size 226, number of requests 2000



(b) OI-instance, fleet size 165, number of requests 1500

Fig. 12. The maximum occupancy of vehicles of on-demand versus traditional bus transport systems.

The maximum URT per direct distance for the TPBT is only 1.45, however the overall average will of course be higher compared to the average of the ODBRP. In addition, it should be noted that in the ODBRP, transfers are not allowed, every request is served by one bus, while in the TPBT the number of transfers is in our instances approximately 53.1% of the total number of requests. This is plotted in Fig. D1. In the simulation, we take no buffer into account to be able to make the transfer and we do not include a penalty cost for making a transfer because we do not want to give any advantages to the ODBRP in the comparison. However, Frei et al. (2017) found in their work that the cost of a transfer amounts to 4.9 equivalent in vehicle minutes (EIVM), while Garcia-Martinez et al. (2018) found a number of 15.2 EIVM for making one transfer. These figures suggest that a transfer has a (perceived) cost in itself, regardless of the delay it causes, and indicates that passengers would prefer a longer direct bus trip above a shorter journey with a transfer.

In this experiment capacity constraints were erased for both systems. To investigate the used capacity for both on-demand and TPBT systems, Fig. 12 plots the maximum occupancy reached and the percentage of the total fleet size reaching this occupancy for both randomly distributed demand and demand from inner to outer city. This chart clearly shows that on-demand buses do not need large vehicles. For the randomly distributed demand (Fig. 12a), there are no on-demand vehicles reaching an occupancy of more than 9 passengers, while the fixed line system still has 28% of the vehicles reaching an occupancy of more than 16. The graph also shows that 73% of all vehicles serve at least four passengers sharing their route, proving that the algorithm finds the ride-sharing opportunities. Occupancies for the TPBT are generally larger because passengers have to transfer, thus using multiple lines. The OI instance (Fig. 12b) is chosen as a second instance to plot, because it has the highest reached occupancy for the on-demand buses with a maximum occupancy of 13. In this instance, there are even more opportunities for ride-sharing, with 70% of the fleet size reaching an occupancy of six passengers, even though the number of requests is lower (1500 instead of 2000).

When the frequency of the TPBT is increased to one bus every 20 minutes, the fleet size necessary to operate the system increases as well. In Table 6, the first effect visible is the decreased total URT for the on-demand system, especially for the larger instances. The columns in bold show the

Table 6
Results: frequency fixed lines 20 and no capacity constraints (in bold the percentage differences in fleet size, on-demand URT, and fixed line URT, compared to a frequency of 30)

	Size			URT					URT ODBRP/direct distance				
Instance	n	b	$\%\Delta_b$	TPBT	$\%\Delta_{\text{TPBT}}$	ODBRP	$\%\Delta_{\text{ODBRP}}$	%Δ	max	Average	Q1	Median	Q2
U	500	261	**+38.8**	30,420	**−1.4**	23,844	**+1.0**	−21.6	1.05	1	1	1	1
T	500	157	**+31.7**	23,340	**−2.5**	17,613	**−0.9**	−24.5	1	1	1	1	1
II	500	68	**+31.4**	11,650	**−1.0**	8,627	**−1.6**	−25.9	1.35	1.01	1	1	1
IO	500	157	**+16.3**	23,340	**−1.1**	17,613	**−0.8**	−24.5	1	1	1	1	1
OI	500	154	**+14.9**	22,450	**−1.5**	17,239	**−1.3**	−23.2	1.1	1	1	1	1
OO	500	248	**+29.1**	33,420	**−1.1**	25,944	**+1.0**	−22.3	1.07	1	1	1	1
U	1000	294	**+40**	59,890	**−1.2**	23,844	**−2.2**	−21.6	1.05	1	1	1	1
T	1000	213	**+34.8**	39,570	**−1.7**	31,042	**−4.9**	−21.5	1.08	1	1	1	1
II	1000	74	**+33.7**	23,390	**−1.0**	/	**/**	/	/	/	/	/	/
IO	1000	179	**+12.6**	46,660	**−1.3**	37,376	**−2.9**	−19.9	1.54	1.03	1	1	1.03
OI	1000	172	**+16.2**	45,740	**−1.2**	38,583	**−6.4**	−15.6	1.61	1.06	1	1	1.08
OO	1000	289	**+38.9**	64,870	**−1.1**	50,028	**−2.3**	−22.9	1.01	1	1	1	1
U	1500	307	**+40.8**	89,780	**−1.4**	70,493	**−7.3**	−21.4	1.65	1.01	1	1	1
T	1500	228	**+38.2**	59,690	**−1.8**	49,366	**−12.6**	−17.3	1.96	1.02	1	1	1.01
II	1500	78	**+29.8**	34,870	**−1.1**	/	**/**	/	/	/	/	/	
IO	1500	189	**+8.0**	69,940	**−1.4**	52,148	**−4.2**	−11.1	2.70	1.09	1	1.03	1.11
OI	1500	187	**+13.3**	68,350	**−1.2**	64,806	**−7.4**	−5.2	2.66	1.13	1	1.07	1.2
OO	1500	307	**+38.2**	97,320	**−1.2**	76,478	**−6.3**	−21.4	1.35	1.01	1	1	1
U	2000	315	**+39.3**	119,350	**−1.3**	97,807	**−10.4**	−18.1	1.68	1.03	1	1	1.04
T	2000	236	**+37.2**	79,030	**−1.8**	70,338	**/**	−11.0	2.61	1.08	1	1	1.09
II	2000	102	**+2.7**	42,070	**−1.0**	/	**/**	/	/	/	/	/	/
IO	2000	192	**+7.3**	92,800	**−1.4**	94,426	**/**	+1.7	3.65	1.21	1	1.10	1.29
OI	2000	195	**+9.5**	92,060	**−1.1**	/	**/**	/	/	/	/	/	/
OO	2000	318	**+40.1**	130,000	**−1.1**	105,884	**−9.0**	−18.6	1.74	1.03	1	1	1.03

percentage difference in fleet size, and URT of both systems compared to a frequency of 30. The total URT of the TPBT decreases slightly over all instances as passengers have to wait a smaller amount of time at their transfer station. The decline in URT for the on-demand system can be explained by the fact that the fleet size is larger when a frequency of 20 minutes needs to be maintained. For an honest comparison, the fleet size used in the on-demand system is equal to the exact amount of buses used in the TPBT system. The larger the fleet size, the larger the flexibility for the routing of the on-demand buses, which is especially important when the number of requests is large. This figure combined with the knowledge drawn from Fig. 12 indicates that an on-demand systems perform better when using a large amount of small vehicles.

Finally, the minimum fleet size necessary to serve all requests in the on-demand system for all instances, when the total number of requests is 2000, is found. In addition, the capacity constraint is readded only for the ODBRP and set to 8. The results can be found in Table 7. It is once again clear that an on-demand system cannot replace efficient fixed lines in an extremely high demand area. Even when the fleet size is increased by 78%, the URT of the on-demand system is 22.1% higher than the URT of the TPBT system. For uniformly distributed demand and demand from outer to

Table 7
Results ODBRP with minimum fleet size serving all requests and capacity 8 versus TPBT without capacity constraints and frequency 30

		ODBRP		TPBT			
Instance	*n*	*b*	URT	*b*	URT	Δ%*b*	Δ%URT
U	2000	218	111,717	226	120,910	−3.5	−7.6
T	2000	202	78,373	172	80,520	+17.4	−2.6
II	2000	143	51,883	80	42,500	+78	+22.1
IO	2000	220	87,192	179	94,140	+22.9	−7.3
OI	2000	235	87,653	178	93,080	+32.0	−5.8
OO	2000	218	119,522	226	131,570	−3.5	−9.2

outer city, comparable with suburban demand, an on-demand system is more efficient. It needs less buses, even with the capacity constraint of 8, and the total URT is 7.6% and 9.2% lower. For the other instances, the trade-off needs to be made between the slight increase in fleet size and decrease in URT. Here, it should be noted that low-capacity vehicles, needed for the on-demand system, will probably cost less compared to the high-capacity vehicles used in the TPBT. Additionally, the total URT obtained with the on-demand system decreases fast when adding a few more vehicles than necessary to serve every request. Figure D2 supports this statement.

5. Conclusions

This paper introduced the ODBRP to support the routing of on-demand buses in an urban environment. An on-demand bus system is a logical next step in the evolution of public transport, as FUT becomes more popular and important. A straightforward LNS heuristic is proposed to solve the ODBRP, minimizing the total URT. BSA increases the flexibility of the routing, adding a new layer of complexity to the optimization problem. The LNS heuristic is used to compare the ODBRP with a traditional public bus system, where lines and timetables are fixed. In general total URT is lower when using on-demand buses instead of the traditional system and transfers are completely avoided. An on-demand bus system is therefore shown to yield shorter passenger trips than a traditional bus system. Only when demand is extremely high, fixed lines are the best way to operate. Additionally our experiments offer some conclusions that might prove useful in the design of a fully on-demand public bus system: our results indicate that the performance of such a system increases when the fleet size is large and the capacity of the buses is rather small.

Future research is needed to expand the static ODBRP to the dynamic version of the problem where only a fraction of the requests are known beforehand. In this case, new requests will pop up during the planning horizon. Bus routes will change while some passengers are already scheduled on them. In the LNS heuristic, when clearing some of the routes, a passenger who is already on the bus at a certain time, cannot be cleared off the schedule, which will lower the solution possibilities. The algorithm will need some adaptations with the aim of lowering the computation time. Second, stochastic events can be taken into account, for example, route blocks or traffic jams. Finally, this study has presented a rather straightforward heuristic for the ODBRP, which leaves ample opportunities for studies into more complicated (and therefore potentially more effective) ways to tackle this problem.

Acknowledgment

This research was supported by a grant of the Research Foundation - Flanders (FWO Strategic Basic Research Fellowship grant).

References

Adawiyah, R., Satria, Y., Burhan, H., 2019. Application of large neighborhood search method in solving a dynamic dial-a-ride problem with money as an incentive. Journal of Physics: Conference Series, Vol. 1218. IOP Publishing, Bristol.

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.

Alonso-González, M., Liu, T., Cats, O., Van Oort, N., Hoogendoorn, S., 2018. The potential of demand-responsive transport as a complement to public transport: an assessment framework and an empirical evaluation. *Transportation Research Record* 2672, 8, 879–889.

Archetti, C., Speranza, M.G., Weyland, D., 2018. A simulation study of an on-demand transportation system. *International Transactions in Operational Research* 25, 4, 1137–1161.

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

Brevet, D., Duhamel, C., Iori, M., Lacomme, P., 2019. A dial-a-ride problem using private vehicles and alternative nodes. *Journal on Vehicle Routing Algorithms* 2, 1, 89–107.

Cordeau, J., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research* 153, 1, 29–46.

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.

Dahle, L., Andersson, H., Christiansen, M., Speranza, M.G., 2019. The pickup and delivery problem with time windows and occasional drivers. *Computers & Operations Research* 109, 122–133.

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. *European Journal of Operational Research* 54, 1, 7–22.

Ellegood, W., Solomon, S., North, J., Campbell, J., 2019. School bus routing problem: contemporary trends and research directions. *Omega* 95, 2020, 102056.

Enoch, M., Potter, S., Parkhurst, G., Smith, M., 2006. Why do demand responsive transport systems fail? Paper presented at Transportation Research Board 85th Annual Meeting. Unpublished.

Errico, F., Crainic, T.G., Malucelli, F., Nonato, M., 2013. A survey on planning semi-flexible transit systems: methodological issues and a unifying framework. *Transportation Research Part C: Emerging Technologies* 36, 324–338.

Finn, B., 2012. Towards large-scale flexible transport services: a practical perspective from the domain of paratransit. *Research in Transportation Business & Management* 3, 39–49.

Frei, C., Hyland, M., Mahmassani, H., 2017. Flexing service schedules: assessing the potential for demand-adaptive hybrid transit via a stated preference approach. *Transportation Research Part C: Emerging Technologies* 76, 71–89.

Fügenschuh, A., 2009. Solving a school bus scheduling problem with integer programming. *European Journal of Operational Research* 193, 3, 867–884.

Garcia-Martinez, A., Cascajo, R., Jara-Diaz, S., Chowdhury, S., Monzon, A., 2018. Transfer penalties in multimodal public transport networks. *Transportation Research Part A: Policy and Practice* 114, 52–66.

Gschwind, T., Drexl, M., 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science* 53, 2, 480–491.

Jokinen, J., Sihvola, T., Hyytiä, E., Sulonen, R., 2011. Why urban mass demand responsive transport? 2011 IEEE Forum on Integrated and Sustainable Transportation System (FISTS). IEEE, Piscataway, NJ, pp. 317–322.

Jung, J., Jayakrishnan, R., Park, J.Y., 2016. Dynamic shared-taxi dispatch algorithm with hybrid-simulated annealing. *Computer-Aided Civil and Infrastructure Engineering* 31, 4, 275–291.

Katzev, R., 2003. Car sharing: a new approach to urban transportation problems. *Analyses of Social Issues and Public Policy* 3, 1, 65–86.

Kim, B.I., Kim, S., Park, J., 2012. A school bus scheduling problem. *European Journal of Operational Research* 218, 2, 577–585.

LocalSolver, 2020a. Optimization solver and services. Available at https://www.localsolver.com/ (accessed 15 April 2020).

LocalSolver, 2020b. Pick up and delivery with time windows. Available at https://www.localsolver.com/docs/last/exampletour/pdptw.html (accessed 12 May 2020).

Mageean, J., Nelson, J., 2003. The evaluation of demand responsive transport services in Europe. *Journal of Transport Geography* 11, 4, 255–270.

Masson, R., Lehuédé, F., Péton, O., 2014. The dial-a-ride problem with transfers. *Computers & Operations Research* 41, 12–23.

Molenbruch, Y., Braekers, K., Caris, A., 2017. Typology and literature review for dial-a-ride problems. *Annals of Operations Research* 259, 1–2, 295–325.

Mulley, C., Nelson, J., 2009. Flexible transport services: a new market opportunity for public transport. *Research in Transportation Economics* 25, 1, 39–45.

Nelson, J., Wright, S., Masson, B., Ambrosino, G., Naniopoulos, A., 2010. Recent developments in flexible transport services. *Research in Transportation Economics* 29, 1, 243–248.

Park, J., Kim, B.I., 2010. The school bus routing problem: a review. *European Journal of Operational Research* 202, 2, 311–319.

Parragh, S.N., Doerner, K.F., Hartl, R.F., Gandibleux, X., 2009. A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks: An International Journal* 54, 4, 227–242.

Parragh, S.N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research* 40, 1, 490–497.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* 40, 4, 455–472.

Santos, D.O., Xavier, E.C., 2015. Taxi and ride sharing: a dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications* 42, 19, 6728–6737.

Savelsbergh, M.W., 1992. The vehicle routing problem with time windows: minimizing route duration. *ORSA Journal on Computing* 4, 2, 146–154.

Schittekat, P., Kinable, J., Sörensen, K., Sevaux, M., Spieksma, F., Springael, J., 2013. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research* 229, 2, 518–528.

Stott, N., 2020. Personal communication.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research* 152, 1, 148–158.

Tellez, O., Vercraene, S., Lehuédé, F., Péton, O., Monteiro, T., 2018. The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity. *Transportation Research Part C: Emerging Technologies* 91, 99–123.

Uber, 2019a. How it works. Available at https://www.uber.com/be/nl/ride/how-it-works/ (accessed 12 February 2019).

Vergeylen, N., Sörensen, K., Vansteenwegen, P., 2020. Large neighborhood search for the bike request scheduling problem. *International Transactions in Operational Research* 27, 6, 2695–2714.

Vlaams Parlement, 2014. De Lijn: Actualisatie van de internationale benchmarkstudie openbaar vervoer. Eindrapport. Available at http://docs.vlaamsparlement.be/pfile?id=1074230 (accessed 19 November 2018).

Winter, K., Cats, O., Correia, G., van Arem, B., 2018. Performance analysis and fleet requirements of automated demand-responsive transport systems as an urban public transport service. *International Journal of Transportation Science and Technology* 7, 2, 151–167.

Yu, X., Miao, H., Bayram, A., Yu, M., Chen, X., 2021. Optimal routing of multimodal mobility systems with ride-sharing. *International Transactions in Operational Research* 28, 3, 1164–1189.

Appendix A: Mathematical model

Table A1
Variables and parameters of the ODBRP

Discrete decision variables	
x_{snb}	1 if the *n*th stop of bus *b* is bus station *s*, and 0 otherwise
y^u_{pnb}	1 if passenger *p* is picked up at the *n*th stop of bus *b*, and 0 otherwise
y^o_{pnb}	1 if passenger *p* is dropped off at the *n*th stop of bus *b*, and 0 otherwise
q_{nb}	Net number of passengers picked up (or dropped off) at the *n*th stop of bus *b*
Continuous decision variables	
t^a_{nb}	Arrival time of bus *b* at its *n*th stop
t^d_{nb}	Departure time of bus *b* at its *n*th stop
T_p	Uride time of passenger *p*
Parameters	
B	The fleet of buses
P	The set of transportation requests, $\|P\|$ denotes the number of requests
S	The set of bus stations
C_b	Capacity of bus *b*
a^u_{ps}	1 if passenger *p* can be assigned to stop *s* for pickup
a^o_{ps}	1 if passenger *p* can be assigned to stop *s* for drop-off
e^u_{ps}	Earliest pickup time for passenger *p*
l^o_{ps}	Latest drop-off time for passenger *p*
$\tau_{ss'}$	Travel time between stop *s* and stop *s'*
N	Positions in the representation of the bus tours, $\|N\|$ denotes the number of positions available

$$\min \text{ URT} = \sum\nolimits_p T_p \tag{A1}$$

s.t.

$$\sum\nolimits_s x_{snb} \leq 1 \quad \forall n \in N, b \in B \tag{A2}$$

$$\sum\nolimits_s x_{snb} = 0 \Rightarrow \sum\nolimits_s x_{s(n+1)b} = 0 \quad \forall n \in N, b \in B \tag{A3}$$

$$\sum\nolimits_s x_{snb} = 0 \Rightarrow \sum\nolimits_p y^u_{pnb} = 0 \quad \forall n \in N, b \in B \tag{A4}$$

$$\sum\nolimits_s x_{snb} = 0 \Rightarrow \sum\nolimits_p y^o_{pnb} = 0 \quad \forall n \in N, b \in B \tag{A5}$$

$$\sum_s x_{snb} = 1 \Rightarrow \sum_p (y^u_{pnb} + y^o_{pnb}) \geq 1 \quad \forall n \in N, b \in B \tag{A6}$$

$$x_{snb} = 1 \wedge y^u_{pnb} = 1 \Rightarrow a^u_{ps} = 1 \quad \forall s \in S, n \in N, p \in P, b \in B \tag{A7}$$

$$x_{snb} = 1 \wedge y^o_{pnb} = 1 \Rightarrow a^o_{ps} = 1 \quad \forall s \in S, n \in N, p \in P, b \in B \tag{A8}$$

$$x_{snb} = 1 \wedge x_{s'(n+1)b} = 1 \Rightarrow t^a_{(n+1)b} \geq t^d_{nb} + \tau_{ss'} \quad \forall s, s' \in S \mid s \neq s', n \in N, b \in B \tag{A9}$$

$$y^u_{pnb} = 1 \Rightarrow t^d_{nb} \geq e^u_{ps} \quad \forall s \in S, p \in P, n \in N, b \in B \tag{A10}$$

$$y^o_{pnb} = 1 \Rightarrow l^o_{ps} \geq t^a_{nb} \quad \forall s \in S, p \in P, n \in N, b \in B \tag{A11}$$

$$\sum_n (ny^u_{pnb} - ny^o_{pnb}) \leq 0 \quad \forall p \in P, b \in B \tag{A12}$$

$$\sum_n (y^u_{pnb} - y^o_{pnb}) = 0 \quad \forall p \in P, b \in B \tag{A13}$$

$$\sum_b \sum_n y^u_{pnb} \leq 1 \quad \forall p \in P \tag{A14}$$

$$x_{snb} + x_{s(n+1)b} \leq 1 \quad \forall s, n, b \tag{A15}$$

$$\sum_p (y^u_{pnb} - y^o_{pnb}) - q_{nb} = 0 \quad \forall n \in N, b \in B \tag{A16}$$

$$\sum_{n' \leq n} q_{nb} \leq C_b \quad \forall n, n' \in N \mid n \geq n', b \in B \tag{A17}$$

$$y^u_{pnb} = 1 \wedge y^o_{pn'b} = 1 \Rightarrow T_p = t^a_{n'b} - t^d_{nb} \quad \forall n, n' \in N \mid n' > n, p \in P, b \in B \tag{A18}$$

$$\sum_p \sum_b \sum_n y^u_{pnb} = |P| \tag{A19}$$

$$x_{snb} \in \{0, 1\} \quad \forall s \in S, n \in N, b \in B \tag{A20}$$

$$y^u_{pnb} \in \{0, 1\} \quad \forall p \in P, n \in N, b \in B \tag{A21}$$

$$y^o_{pnb} \in \{0, 1\} \quad \forall p \in P, n \in N, b \in B \tag{A22}$$

$$q_{nb} \in \mathbb{Z} \quad \forall n \in N, b \in B. \tag{A23}$$

$$\tag{A24}$$

In this formulation, some of the constraints use *if-then* (⇒) and *and* (∧) logical operators, as this is easier to read and understand. However, they can be easily linearized and written as follows:

$$\sum_s (x_{snb} - x_{s(n+1)b}) \geq 0 \quad \forall n \in N, b \in B \tag{A3$'$}$$

$$M \sum_s x_{snb} - \sum_p (y^u_{pnb} + y^o_{pnb}) \geq 0 \quad \forall n \in N, b \in B \tag{A4-A5-A6$'$}$$

$$\sum\nolimits_s x_{snb} - \sum\nolimits_p (y^u_{pnb} + y^o_{pnb}) \le 0 \quad \forall n \in N, b \in B \tag{A4-A5-A6$''$}$$

$$x_{snb} + y^u_{pnb} - a^u_{ps} \le 1 \quad \forall s \in S, n \in N, p \in P, b \in B \tag{A7$'$}$$

$$x_{snb} + y^o_{pnb} - a^o_{ps} \le 1 \quad \forall s \in S, n \in N, p \in P, b \in B \tag{A8$'$}$$

$$t^a_{(n+1)b} - t^d_{nb} - \tau_{s's} + (x_{snb} + x_{s'(n+1)b} - 2)(-M) \ge 0 \quad \forall s, s' \in S \mid s \ne s', n \in N, b \in B \tag{A9$'$}$$

$$t^d_{nb} - e^u_{ps} + (y^u_{pnb} - 1)(-M) \ge 0 \quad \forall s \in S, p \in P, n \in N, b \in B \tag{A10$'$}$$

$$t^a_{nb} - l^o_{ps} + (y^o_{pnb} - 1)M \le 0 \quad \forall s \in S, p \in P, n \in N, b \in B \tag{A11$'$}$$

$$T_p + (2 - y^o_{pn'b} - y^u_{pnb})M - t^a_{n'b} + t^d_{nb} \ge 0 \quad \forall n, n' \in N \mid n' > n, p \in P, b \in B \tag{A18$'$}$$

$$T_p + (2 - y^o_{pn'b} - y^u_{pnb})(-M) - t^a_{n'b} + t^d_{nb} \le 0 \quad \forall n, n' \in N \mid n' > n, p \in P, b \in B. \tag{A18$''$}$$

The objective function (A1) minimizes the total user ride time (URT).

Constraints (A2) enforce the fact that a bus can only stop at one station at the same time.

Constraints (A3) ensure stops that the positions used in the bus route are used consecutively and start at the first position. Constraints (A3′) are the linearized form of constraints (A3).

Constraints (A4) and (A5) impose that no passengers can get on or off the bus at position n, when the bus does not make a stop at position n, while constraints (A6) ensure that when a stop is made, at least one passenger gets either on or off the bus. The linearized way of writing these three groups of constraints can be found in constraints (A4-A5-A6′) and (A4-A5-A6’′).

Bus station assignment is modeled in constraints (A7) and (A8), which enforce that if a bus stops at a certain bus station to pick up (drop off) a passenger, the station has to be in the set of allowed stops for the pickup or drop-off of this passenger. The linearized form can be found in constraints (A7′) and (A8′), respectively.

The time aspect of the ODBRP is incorporated in constraints (A9), (A10), and (A11), or in (A9′), (A10′) and (A11′) in the linearized form. If s is the nth stop of the bus and s' is the $n+1$th stop of the bus, then the time of arrival at s' should be greater than the departure time at s plus the travel distance between stops s and s'. Also, the departure time (arrival time) of the bus at the nth stop is not smaller than the earliest departure time (latest arrival time) of a passenger assigned to that stop.

Constraints (A12), (A13), and (A14) enforce that a passenger is picked up before he is dropped off and that both pickup and arrival are carried out only once by the same bus. For easy reading purposes, constraints (A15) forbid that the same bus station is visited twice in a row.

Constraints (A16) and (A17) enforce the capacity of the buses. The net number of passengers picked up at the nth stop of the bus is equal to the number of passengers picked up minus the number of passengers dropped off at that stop. This variable is used in the last constraint ensuring that the capacity of the buses is not exceeded. Constraints (A18) (or both (A18′) and (A18’′) ln the linearized form) define the variable used in the objective function. Constraints (A18’′) is nonessential when minimizing the total URT, however is included for completeness and necessary if a different objective function would be used. Constraint (A19) ensures that every request is served.

Lastly, constraints (A20), (A21), and (A22) define the binary decision variables, and constraints (A23) enforce the quantity decision variable to be integer.

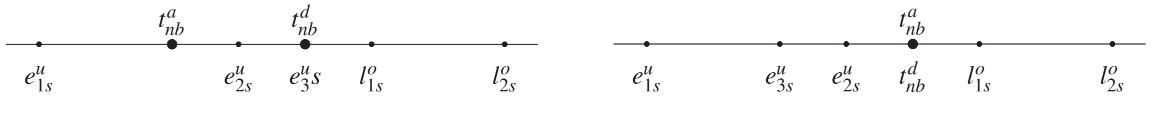
Appendix B: Computational speedup

When naively implemented, the feasibility checks in Algorithm 1 have a computational complexity of $O(n^2)$. In the worst case, when we would have one available bus and we would have planned all the requests in this bus except for one, we have $2n - 2$ insertion positions in a route: for each of the $n - 1$ requests a different origin and arrival stop. When we would want to insert an origin stop for the last request at the beginning of the route, we need to check $2n - 3$ positions further in the route for lateness caused by the detour. To reduce this complexity, a data structure is introduced in which the slack times of all drop-offs are preprocessed. This was initially introduced as the forward time slack by Savelsbergh (1992) for the vehicle routing problem with time windows, but an adapted version was also used by Cordeau and Laporte (2003) in the DARP.

Because the ODBRP has different characteristics from the DARP, some adaptations to this approach are necessary. One of the differences with Cordeau and Laporte (2003) is the use of three time stamps per vertex (or stop). In particular, they use the arrival time of a vehicle at a stop, the beginning of the service at the stop, and the departure time of the vehicle at the stop. In the ODBRP, only two time stamps are used as is explained in Fig. 3: the arrival time of the vehicle at the stop and the departure time of the vehicle at the stop. Although Fig. 3 is correct in theory, the algorithm has been designed in such a way that it will partially optimize this time frame automatically. This is shown in Fig. B1a. By design, the algorithm will ensure that the bus departs as early as possible to the next stop, this means that t^d_{nb} moves to the front of the time line compared to the time line of Fig. 3, and more specifically it moves to the moment when the last scheduled passenger (passenger 2 in the figure) can get on the bus. All passengers who need to get off the bus at station s, do so at time t^a_{nb} and all passengers who have to get on the bus are scheduled at time t^d_{nb}. As the service time is considered negligible, it is currently not included in the algorithm. Therefore, if e^u_2 and e^u_3 would lie before t^a_{nb} on the time line, no waiting time would occur and t^a_{nb} would be equal to t^d_{nb}. This is shown in Fig. B1b. Here the pickup and drop-off events happen at the same time.

Another difference with the DARP is the use of the maximum ride time, which is not necessary in the ODBRP because maximum ride times are incorporated in the time window of each passenger.

The slack time of a certain passenger is calculated by subtracting the actual arrival time of the bus at his destination bus stop from his latest arrival time. When constructing a solution, every available bus has a slack vector of the same size (i.e., with the same number of filled positions) as its route composed until now. In every position of this vector, the minimum of the slack time for



(a) Time aspect at bus stop with waiting in the algorithm (b) Time aspect at bus stop without waiting in the algorithm

Fig. B1. Time aspect at bus stop.

Bus stop	1	2	*	3	4
Bus station	7	3	4	6	11
Arrival time	30	42		52	90
Departure time	30	42		52	90
P on	2	1			
P off				1	2
Slack vector	40	40		40	40

Fig. B2. Toy example—slack vector.

this position and for every position further in the route is represented. The minimum slack time is included because this is the most restrictive when inserting other requests in the route. Assuming that the route is feasible, that is, all passengers arrive at their destination on time, this slack time is always positive or equal to 0. Because the slack vector holds the minimum slack time for the current position and every position further in the route, the content of the slack vector can only become larger or stay equal when iterating over the positions from left to right. For example, at the last position, only the passengers getting off at this stop have to be taken into account, while at the first position, the minimum slack time of all the passengers scheduled in the route is shown. Every time a passenger request is inserted in the solution, the slack vector is updated.

In the route that we obtained in the toy example in Fig. 4d, the slack time of passenger 1 is $100 - 52 = 48$ and the slack time of passenger 2 is $130 - 90 = 40$. As a consequence, the slack vector for this example has four positions (one for every bus stop) and has a value of 40 in every position as the second passenger gets off the bus last and has the smallest slack time. A follow-up of the toy example from Fig. 4d with slack vector is illustrated in Fig. B2. A new request needs to be inserted and we want to check if it would be feasible to insert station 4 after the second stop of the bus route. The delay caused by this insertion equals $-10 + 20 + 35 = 45$ (-10 to delete the connection between stations 3 and 6, 20 for the distance between stations 3 and 4, and 35 for the distance between stations 4 and 6). Using the slack vector, it can be immediately discovered that this insertion would be infeasible, without checking the latest arrival times of every passenger drop-off later on in the bus route.

Appendix C: Algorithm quality assessment

C.1. Localsolver

To assess the quality of our algorithm, we first compare our results with the results of the heuristic software LocalSolver (LocalSolver, 2020a). LocalSolver is a global optimization solver that uses both exact and heuristic methods. Since pure exact solvers are incapable of solving anything but the smallest of instances, this is a more meaningful alternative. Our implementation is based on the pickup and delivery problem with time windows (LocalSolver, 2020b), with the necessary adaptations to include bus station assignment. Our implementation was checked and approved by the developers at LocalSolver (N. Stott,personal communication). Figure C1 shows the comparison in computation time and solution quality for an instance with 500 requests. If every request would be served by a different bus, thus if every passenger would have a direct taxi ride from origin to

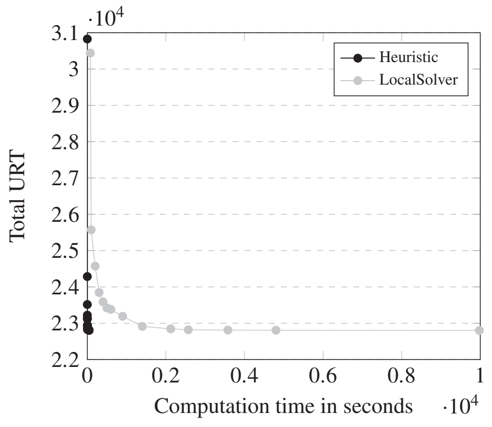


Fig. C1. Comparison LocalSolver (121 stations, 500 requests, 182 buses with capacity 8).

destination, the total URT of the instance would be 22,802. This requires a fleet size of 182 buses with a capacity of 8 people. With this fleet size, LocalSolver finds a first feasible solution after 77 seconds with a total URT of 33,442. The solution quality keeps improving until a total URT of 22,802 is found after 9977 seconds. This is the optimal solution because obviously, the total URT cannot be smaller than in the situation where all passengers are transported without delay from their origin to their destination. Our algorithm, on the other hand, finds an initial solution with a total URT of 30,827 in 0.10 seconds, which quickly improves to 23,119 within one second. After 46 seconds, our algorithm also finds the optimal solution.

To further investigate the algorithm's quality, we solved several larger instances in LocalSolver with a tighter fleet size. An instance of 1000 randomly distributed requests was solved with a fleet size of 221 buses with capacity 8. The time limit for LocalSolver was set at 24 hours. The best URT found was 48,757. Our heuristic algorithm solved the instance three separate times with a limit of 1000 iterations. On average, after 386 iterations the last improvement in URT was seen, requiring a computation time of approximately 120 seconds. The gap in solution quality was 0.3% on average in favor of LocalSolver. Using our algorithm within the first second, the solution improves by 19%, after five seconds the improvement is 22%, and after two minutes 22.9%. This indicates that once again our algorithm finds a qualitative solution in a reasonable amount of time. Localsolver was not able to solve instance of 2000 requests within a week of computing time.

C.2. DARP benchmark instances

All things considered, when compared to existing models in the literature, the ODBRP is closest to the DARP. In this section we will run our algorithm on modified instances of the DARP to further assess its quality.

First, the benchmark DARP instances of Cordeau and Laporte (2003) are converted to fit the ODBRP. In the $[-10, 10]^2$ square, bus stations are introduced at every point (x, y) , $\forall\, x, y = 2n$ with $n \in \mathbb{Z} : n \in [-10, 10]$. The maximum walking distance is set to 2 minutes, which corresponds to the maximum walking distance of 10 minutes on the 100×100 grid used in the experiments in Section 4. Subsequently, every request is assigned a set of possible stations for origin and arrival within a two-minute walking distance from their actual origin and arrival location. Requests for a ride with a direct distance smaller than four minutes (two times the walking distance) are set aside, as these potential passengers can directly walk from their origin to their destination. As mentioned in Section 1, the DARP generally has a separate maximum ride time constraint and time windows on both pick-pickup and drop-off events, while the ODBRP has only one time window within which the transportation needs to take place, and which also defines the maximum ride time for each user. The benchmark DARP instances of both Cordeau and Laporte (2003) include inbound and outbound requests. The inbound (outbound) requests have a restricted time window $[e_i, l_i]$ on the drop-off (pickup) event and a nonrestricted time window $[0, T]$, where T equals the end of the planning horizon, on the pickup (drop-off) event. The maximum ride time is 90 minutes. Consequently, for every inbound (outbound) request i, the single time window used in the ODBRP becomes $[e_i - 90, l_i]$ ($[e_i, l_i + 90]$). As in the benchmark DARP instances, a service time of 10 minutes per stop is adopted and the capacity of the buses is set to 6. Instances 1–10 have smaller time windows compared to instances 11–20.

In Cordeau and Laporte (2003), the total cost (TC) is minimized, which equals the total traveled distance (TTD) of the vehicles. To compare the TC of both DARP and ODBRP, our algorithm for the ODBRP needs some adaptations. In fact, the MaxP constructive operator minimizes the TTD, as it inserts a request where the detour is smallest. Ergo, this becomes the sole constructive operator in this version of the ODBRP. Furthermore, the first clearing criterion of the LNS when minimizing the URT is changed from clearing the bus with the largest URT, to clearing the bus with the largest distance traveled. The other clearing criteria are kept. In the LS part, requests are relocated to positions where the TTD decreases the most. Above this, a maximum route duration of 480 minutes is adopted as an extra constraint, just like in the DARP, and a depot is introduced. In the DARP, every bus route starts and ends at the depot. Therefore the depot is also added to the ODBRP, but after optimizing the routes. The location of the depot is presumed equal to the location of the depot in the benchmark DARP instances. In Table C1, the best known solutions for the DARP (Braekers et al., 2014) are depicted next to the costs obtained with the ODBRP. For all instances except for one, the ODBRP obtains lower costs. Percentage differences in TC between the DARP and the ODBRP can be found in the gray column. This can be attributed to bus station assignment and passenger pooling. Instances in which passengers have more clustered demand will perform better in the ODBRP, which is not the case in instance pr10. Another advantage of bus station assignment is the lower total route duration obtained. For example for instance pr03, Cordeau and Laporte (2003) find a total route duration of 5056.83, while we find one of 3739. These large differences in total route duration are mostly due to a smaller number of stops in a route, resulting in less total service time.

In the most right part of Table C1, the results of the "normal" ODBRP when minimizing the total URT are demonstrated. In this case, the maximum route duration constraint of 480 minutes is absent. The depot is kept to calculate the TC. Compared to when the cost is minimized, URTs are 9.5–87.6% lower. For a public bus transportation system to be appealing to the passengers, this is the most important factor. These results consequently indicate the importance to introduce the ODBRP as a new optimization problem for on-demand bus systems. Especially when these buses will be autonomous, the cost is of lesser importance and also the maximum route duration

Table C1
Comparison DARP and ODBRP: minimizing the total cost and minimizing total URT (n_w is the number of passengers that are able to walk directly to their destination)

Instance			DARP		ODBRP (min. TC)			ODBRP (min. URT)		
Name	n	b	TC	n_w	TC	Δ_{TC}%	URT	TC	URT	Δ_{URT}%
pr01	24	3	190.02	7	132	−30.8	461	204.8	165	−64.2
pr02	48	5	301.34	12	223.2	−26.1	982	363.4	309	−68.5
pr03	72	7	532	6	481.4	−9.3	1934	748	870	−55
pr04	96	9	570.25	12	536.2	−6.4	2715	664	2362	−13
pr05	120	11	626.93	22	545.2	−13.1	2715	768.4	2157	−20.5
pr06	144	13	785.26	21	706.6	−10	3568	987.4	2540	−28.8
pr07	36	4	291.71	7	208.4	−28.6	787	296.6	712	−9.5
pr08	72	6	487.84	10	423.8	−13.1	2249	648.4	950	−57.7
pr09	108	8	658.31	15	596.2	−9.4	3101	893.6	1715	−44.7
pr10	144	10	851.83	16	864.4	+1.5	3483	1032.8	2930	−15.8
pr11	24	3	164.46	7	117	−28.8	856	214	106	−87.6
pr12	48	5	295.66	12	234.8	−20.6	1081	344.6	476	−56
pr13	72	7	484.83	6	458	−5.5	2260	727	1041	−53.9
pr14	96	9	529.33	12	475	−10.2	2762	654	1972	−28.6
pr15	120	11	577.29	22	526.8	−8.7	3561	762.8	2791	−21.6
pr16	144	13	730.67	21	696.8	−4.6	4390	1099.2	2388	−45.6
pr17	36	4	248.21	7	190.8	−23.1	1138	286.8	803	−29.4
pr18	72	6	458.73	10	433.6	−5.4	2587	640.2	1502	−41.9
pr19	108	8	593.49	15	574	−3.3	3282	640.2	1880	−42.7
pr20	144	10	785.68	16	783.4	−0.3	4675	1096	2472	−47.1

constraint is only fitted when drivers are present. Even though the maximum route duration constraint is absent, the average maximum route duration for these instances in the "normal" ODBRP amounts 533 minutes, which is still moderate and workable.

Appendix D: Supporting material

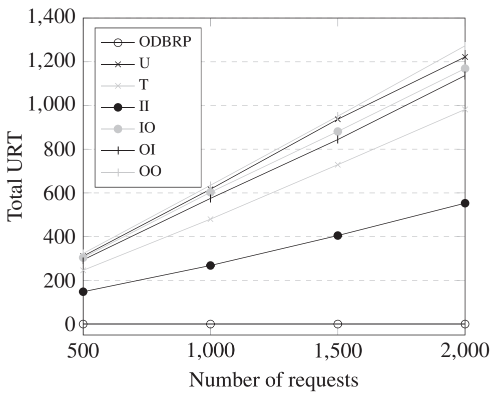


Fig. D1. Transfers TPBT versus ODBRP.

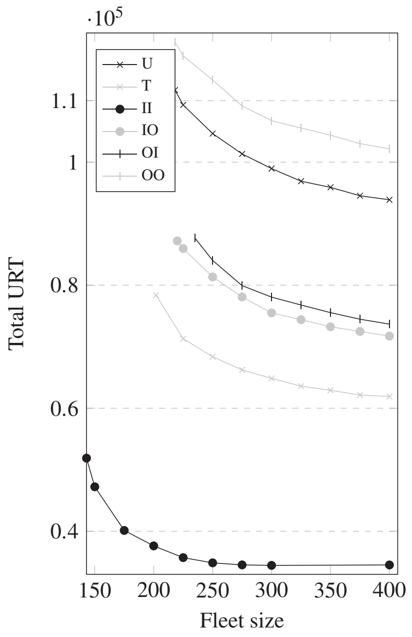


Fig. D2. Effect fleet size on URT ODBRP for different demand patterns.