University of Antwerp
Faculty of Business and Economics

DEPARTMENT OF ENGINEERING MANAGEMENT

The On-Demand Bus Routing Problem with Real-Time Traffic Information

Ying Lian, Flavien Lucas & Kenneth Sörensen

UNIVERSITY OF ANTWERP
Faculty of Business and Economics
City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
www.uantwerpen.be

FACULTY OF BUSINESS AND ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

The On-Demand Bus Routing Problem with Real-Time Traffic Information

Ying Lian, Flavien Lucas & Kenneth Sörensen

RESEARCH PAPER 2022-003
MAY 2022

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
e-mail: joeri.nys@uantwerpen.be

The research papers from the Faculty of Business and Economics are also available at www.repec.org (Research Papers in Economics - RePEc)

D/2022/1169/003

The On-Demand Bus Routing Problem with Real-Time Traffic Information

Ying Lian, Flavien Lucas, Kenneth Sörensen

ANT/OR - Operations Research Group, Department of Engineering Management, University of Antwerp

Abstract

We propose to solve a real-time traffic variation of the On-Demand Bus Routing Problem (ODBRP) introduced by Melis and Sörensen (2021). The ODBRP belongs to the category of dial-a-ride problems (DARP), and features departure and arrival bus station selection. This problem is specifically aimed at planning a fleet of on-demand buses in an urban environment. However, cities are frequently plagued by traffic congestion, which may cause delays and missed time windows for passengers.

To deal with this situation, we introduce, study, and solve a variant of the ODBRP in which travel times are both time-dependent (i.e., the travel time between two nodes depends on the departure time) and updated dynamically.

In our approach, congested roads that might cause passenger delays are modeled by frequently updating the travel speed on the road segments that constitute them. The resulting problem is solved by using a K-shortest paths procedure to determine alternative paths between bus stations combined with a variable neighborhood search procedure to repair violated time windows.

Our experimental results show the overall effectiveness of this real-time control under different degrees of flexibility (congestion and number of buses available). Specifically, the average tardiness, maximum tardiness, and number of late passengers are significantly reduced under a wide range of congestion scenarios, from slight to severe. In addition, this effectiveness holds for various ratios of requests to the number of vehicles.

Keywords: on-demand bus, real-time traffic information, dynamic travel speed, real-time control

1. Introduction

In most routing problems, the travel time on the edges between nodes is considered to be static. In practice, however, driving speeds are subject to change throughout the day: a car is slower during rush hour than at midday. Events on the road, such as accidents or road renovations also have an impact on the average speed of the road. Malandraki and Daskin (1992) explain the evolution of speed on the road by two types of perturbations: the first due to traffic volume and the second due to random events.

Speed is particularly impacted by the traffic volume, which varies throughout the day. To take this into account, some papers calculate the travel time on each edge using a function based on both the distance travelled, and the departure time. A common way to model such *time-dependent* travel times is to divide a day into 24 hours, and allow for 24 different travel times on the edge, depending on the hour during which travel on the edge begins. More details about time-dependent problems are given in section 2.1.

Traffic speed also fluctuates due to random events, such as weather conditions and accidents and is therefore generally also stochastic. Many papers (Fu, 1999, 2002; Taş et al., 2013, 2014) have introduced stochastic travel speeds in routing problems and use the expected speed values to improve the reliability of the solution. However, stochastic problems are still difficult to solve and speed prediction can be inaccurate. Methods solving stochastic problems are also generally too time-consuming to be used in a real-time context.

Fixed speed problems may consider only one edge for each pair of nodes, representing their shortest path. However, stochastic or time-dependent problems can be improved by explicitly calculating the shortest path using multigraphs (Huang et al., 2017). For example, Garaix et al., 2010; Setak et al., 2015 use multigraphs

to solve some variation of the VRP in the planning period while Huang et al. (2017) use it for real-time adaptation of the vehicle routes for stochastic problems.

In this paper, we study how multigraphs can help adapt vehicle routes in unexpectedly congested situations. Specifically, this paper is based on the static ODBRP (Melis and Sörensen, 2021) and adds an approach to adapt the solution in real time to the actual travel speeds.

The paper is structured as follows. Section 2 presents a review of the literature on the most relevant topics. The problem studied is described in Section 3, followed by an explanation of the solution approach in Section 4. Then, both the procedure to generate an initial solution and the method developed to adapt it to dynamic events are explained in Section 5 and tested in Section 6, coupled with analyses of the results. The paper concludes and discusses possible future research in Section 7.

2. Literature Review

This paper builds on the work of Melis and Sörensen (2021), who introduce the on-demand bus routing problem (ODBRP). In the ODBRP, a set of passenger requests to travel from an origin location to a destination location must be fulfilled by a set of buses. Each request is associated with a time window, delimited by the earliest departure time at the origin and the latest arrival time at the destination. All requests are known in advance and each origin and destination is associated with a set of bus stations that can be used to pick up or drop off the customer. The aim of the ODBRP is to minimize the total user ride time (i.e., the sum of all individual user ride times) by assigning requests to buses, planning bus routes and scheduling arrival and departure times at each station.

The ODBRP is closely related to the dial-a-ride problem (DARP), a routing problem dealing with door-to-door passenger transport. Different from the DARP, however, passengers are not picked up at their origin location but asked to walk to a nearby bus station (we will use the term "(bus) station" to refer to the physical location where passengers can be picked up or dropped off and reserve the term "(bus) stop" for the action of stopping the bus to pick up or drop off passengers). Similarly, they are dropped off at a bus station close to their destination, from where they can continue to their destination on foot. The main motivation for this feature is that, in many urban environments, buses can only stop at predefined and clearly signposted locations. In other words, the main distinction between the ODBRP and the DARP is the possibility to select the pick-up and drop-off station for each passenger from a set of potential stations, that are all within walking distance of the passenger's origin or destination.

This paper discusses an extension of the ODBRP, with time-dependent speed and real-time traffic conditions. Further details on the ODBRP are available in Melis and Sörensen (2021), and a recent review on the DARP can be found in Ho et al. (2018).

In the next paragraphs we discuss the related literature concerning time-dependent routing problem (TDVRP), shortest path calculation, and real-time control in transport and vehicle routing problems.

2.1. Time-dependent routing

Problems defined as "time-dependent" consider the travelling time from one node to another not only proportional to its geographical distance, but also dependent on the departure time. The first study on the TDVRP is carried out by Malandraki and Daskin (1992), who treat travel time as a step function depending on the departure time. To solve this problem, the authors propose heuristics based on nearest neighbor and cutting plane methods. Hill and Benton (1992) propose procedures to estimate time-dependent travel speed parameters and hence solve the problem of data collection and storage. A few years later, Malandraki and Dial (1996) uses a restricted dynamic programming heuristic to solve time-dependent travelling salesman problem.

Early papers like the ones mentioned above do not take into account the *first in, first out* (FIFO) property, which can lead to the counter-intuitive situation where a vehicle arrives later because it has departed earlier or vice versa. The FIFO property is introduced for the first time by Ichoua et al. (2003) to solve the TDVRP with soft time windows. The authors solve this problem using a parallel tabu search heuristic, and also reveal

the better performance of time-dependent over constant speeds. In addition, they introduce the concept of step functions to define travel speed, transforming travel time into a piece-wise linear function.

In a later contribution, Fleischmann et al. (2004a) derive raw data from a traffic center and model travel times as a smoothed piece-wise linear function, thus avoiding jumps in travel times and violation of the FIFO property. Maden et al. (2010) perform a case study of the VRPTW with time-dependent travel speeds, solved by tabu search while Kok et al. (2012) solve the TDVRP with a modified Dijkstra's algorithm and a restricted dynamic programming method. Recently, Ho et al. (2018) estimate travel times using staircase regression of discrete speed data to solve the DARP. Finally, Rincon-Garcia et al. (2017) solve the TDVRP with hard time windows with a combination of large neighborhood search and variable neighborhood search while Spliet et al. (2018) use a branch-and-price-and-cut algorithm.

For a more thorough review on the TDVRP we refer to Gendreau et al. (2015).

2.2. Shortest Path Updating for Routing Problems

The classic TDVRP, which minimises travel time, lays the foundation for shortest path updating. As the travel time varies with time intervals, the shortest path between each pair of nodes may also vary between each time period.

This problem is modeled as the "time-dependent quickest path problem" (TD-QPP) in Gendreau et al. (2015). Dehne et al. (2012) prove that the calculation of all possible start times from a specific node is NP-Hard. However, the TD-QPP with FIFO property is polynomially solvable, as shown in Kaufman and Smith (1993), since the travel time functions are known. Fleischmann et al. (2004a) emphasize the potential value of taking shortest path recalculation into consideration. Work has also been done to add realism, such as Eglese et al. (2006) who convert a real world map into a realistic time-dependent graph. More recently, Garaix et al. (2010) consider the road network as a multigraph, which turns the path selection problem into a fixed sequence arc selection problem, equivalent to the multidimensional multiple choice knapsack problem (MMKP). However, in this paper, travel speeds on arcs are deterministic and time-independent. Setak et al. (2015) add time-dependency to the path selection problem by dividing travel time into intervals for each edge. This problem is still deterministic, thus allowing the authors to solve it with a classical search method such as tabu search.

In parallel, Kim et al. (2016) propose a realistic method for estimating the transition probabilities of traffic congestion states and arc travel time distributions in the transportation network. Their model is built on deterministic travel speeds, and they does not include stochastic future fluctuations.Huang et al. (2017) also use piece-wise constant travel speeds and a modified Dijkstra algorithm for the time-dependent shortest path, but extend it for real-time adjustment using path selection as a recourse action. Finally, Vidal et al. (2021) uses dynamic programming to calculate time-dependent shortest path with continuous speed in a closed form, without approximation or discretization.

2.3. Real-time Control in Routing Problems

Public transport planning typically happens on several levels, spanning tactical, strategical, operational, and real-time decisions (Ibarra-Rojas et al., 2015). This paper will contribute to the study of the last stage in the planning process, real-time control, the only one operating when the vehicles have started travelling.

With the technological development of tools such as Automatic Vehicle Location Systems (AVL), Geographic Positioning Systems (GPS), Automatic Passenger Counting Systems (APC) and Advanced Transportation Management Systems (ATMS) (Anil Rao et al., 2015), decision-makers are able to know the real state of public transportation, thus can deal with unexpected variations and implement real-time control operations to improve service efficiency. However, most implementations are limited to in-station control (for example, by keeping vehicles waiting at specific stations) and inter-station control (by studying travel speed and road traffic).

This paper focuses mainly on strategies related to real-time route and vehicle schedule changes, with or without reassignment of customers/passengers, using up-to-date traffic information. Online traffic information is first applied to the dynamic vehicle routing problem by Fleischmann et al. (2004b), who introduce a planning framework for the dynamic calculation of the shortest path. Cortés et al. (2008) proposes a hybrid

adaptive predictive control method for the dynamic pickup and delivery problem (DPDP), which takes into account both predictable and unpredictable congestion. To solve their DARP with dynamic request, Schilde et al. (2014) reschedules the vehicle routes based on the updated travel speed revealed when the vehicle departs and then treats dynamics request with a VNS.

In their paper, Ibarra-Rojas et al. (2015) solve a pick-up and delivery problem with dynamic requests using local search and a diversification strategy. The authors also integrate traffic jams (considered as a driver-estimated waiting time) and fluctuations in travel times (modelled by mean and variance).

Huang et al. (2017) uses two-stages stochastic mixed integer program: the first stage solves the time-dependent vehicle routing problem with stochastic programming and models the uncertain state of the traffic using the expected value and variance. The second stage calculates and selects the shortest path as a recourse action when the traffic pattern is known.

3. Problem Description

A two-stage passenger transportation problem with dynamic traffic information is discussed in this paper. In the first stage, a bus routing and scheduling, passenger-bus assignment is determined, as well as bus station assignment. As a result, passengers are planned to be transported with the minimum total URT, based on historical traffic information. In the second stage, the solution is improved in real time according to gradually revealed true traffic information.

For the purpose of this paper, we assume an additional network layer underlying the graph that connects the stations. This network layer corresponds to the road network and contains crossings and road segments, where a road segment is defined as the part of route between two consecutive crossings. Let d_{ij} be a path between any two nodes i and j in a road network. Then the time to travel d_{ij} if depart at time t is $TT_{ij}(t) = \overline{TT_{ij}}(t) + TT_{ij,stoc}(t)$, where $\overline{TT_{ij}}(t)$ is a time-dependent value that can be calculated given the historical average travel speed and t, thus $\overline{TT_{ij}}(t)$ is known in the first stage. More details of $\overline{TT_{ij}}(t)$ are in section 4. The other term $TT_{ij,stoc}(t)$ is stochastic, and we assume it can be only revealed the moment a bus is upon traveling d_{ij}. More details of $TT_{ij,stoc}(t)$ are in section 4.2. Furthermore, the calculation of shortest paths are explicitly considered in both stages. More specifically, in the first stage, top K shortest paths among all station pairs are calculated based on the time-dependent travel time mentioned above, and they form a candidate pool for each station pair, with the shortest one temporarily chosen. While in the second stage, the path that actually travel can change among these candidates according to the true travel time. These shortest paths are explained in section 4.1.

In the ODBRP, passengers send in travel requests, possibly via a mobile application or website. The requests can be sent in advance, for example, one day before their trip, or in real-time to have a service as soon as possible. Potential passengers also specify their locations of origin and destination, while the buses are only allowed to dwell at predefined stations, so the feasible station(s) within walking distance of each passenger's origin and destination are implicitly calculated. Unlike door-to-door transportation in the standard formulation of DARP, which allows passengers to determine where to board and alight, ODBRP handles bus station assignment with predefined stations, which significantly improves the overall performance of the solution, since these stations are assigned in an optimal or a near-optimal(if heuristic methods are adopted) way.

A more formal description is as following, with the exact mathematical model in Appendix A.

Consider a homogeneous fleet of public on-demand buses, each with a limited capacity of Q. Let St be the set of bus stations available for each bus for pickup and drop-off the passengers. Each station is visited only when needed and can be visited more than once or even never. Buses can only stop at stations, rather than going door-to-door. Passengers' requests are geographically dispersed, each with a location of origin and destination, allowing to list all stations within walking distance for each request. We assume that at least one station for pick-up and one station for drop-off can be reached by each passenger.

Let R be the set of all requests, while for each request $r \in R$, the set of bus stations $\subset St$ available for pick-up/drop-off are noted Set^r_{on} and Set^r_{off}.

Each request also has a time window of earliest allowed departure at the assigned departure station and latest allowed arrival at the assigned arrival station $[e_r, l_r]$. The duration of each time window is set to

$f \times t_{dir}$, where t_{dir} is the direct travel time from the departure station to the arrival station, and f is a constant. As we consider the on-demand bus as a trade-off between the traditional fixed public bus and fully on-demand transport (e.g., taxi and car-pooling), we intuitively set f to 1.6 in this paper. A small value of this parameter leads to a strict time window and vice versa. In this paper, we assume that each request is known in advance and corresponds to only one passenger.

The station sequence assigned to each bus b is denoted SS_b, and its arrival and departure times at the corresponding station s are noted t^a_{sb} and t^d_{sb}. In our model, buses are allowed to wait at a station, so the departure time may be later than the arrival time, even if there are passengers on board. Waiting at a station increases the ride time of passengers already in the bus but can significantly reduce the ride time of several passengers, resulting in a reduction of the total URT. The difference between departure and arrival time at a station s is called *waiting time*: $t^w_{sb} = t^d_{sb} - t^a_{sb}$. This waiting policy is only used if at least one passenger is boarding at that station and the bus arrives before the earliest departure in the passenger's time window. We also assume that passengers are picked up and dropped off immediately when the vehicle arrives, and we omit this service time, assuming that the departure time is simply equal to the arrival time if waiting time is not necessary. Thus, each bus is given a list of the P_b passengers it will be charged with carrying, including their pick-up and drop-off times and locations.

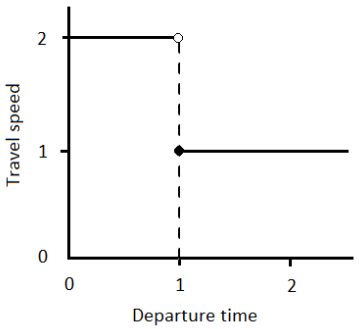
We assume that the route of each bus starts directly at the first passenger's pick-up station and ends at the last passenger's drop-off station and neglect the route to or from the depot. This is not unrealistic, as the travel to and from the depot will be done empty and therefore not contribute to the total URT. Finally, the travel speed from node i to node j at time t is denoted $TS_{ij}(t)$. In addition, the travel time from i to j if a bus leaves on time t, is noted as $TT_{ij}(t)$, where i and j can be either stations or intersections.

The top priority in both stages is to minimize passenger dissatisfaction. In this regard, we adopt different objective functions for the two stages. In the initial stage, we aim to minimize the total URT with all constraints satisfied; while in real-time stage, punctuality is deemed more important than total URT, since congestion can cause violations of time windows. Consequently, the objective function is to minimize the total tardiness of all late passengers. In terms of the constraints, except the latest arrival time l_r of each request, others such as capacity, precedence, coupling and earliest allowed departure time are still considered.

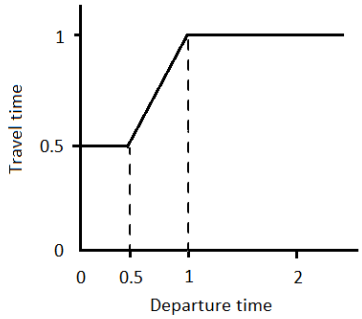
4. Calculation of shortest path and travel time update

This section discusses two procedures that are required to model and solve the real-time ODBRP. The time-dependent shortest path procedure is described in section 4.1. Besides, the generation of the network-consistent stochastic deviation is explained in section 4.2.

Given the definition of time-dependent travel speed explained in section 3, for each road segment ij, the travel speed $TS_{ij}(t)$ is constant in each time interval, but may vary in different time intervals. Mathematically, as proposed by Ichoua et al. (2003), $TS_{ij}(t)$ is a step function of different time intervals (figure 1a); It leads to a piecewise linear travel time function $TT_{ij}(t)$ (figure 1b) and guarantees the "First in, first out" (FIFO) property: when crossing the same road segment, a bus leaving earlier should also arrive earlier.



(a) Step speed function



(b) Piecewise linear travel time function

Figure 1: Travel speed and travel time

Travel time between stations can be calculated by determining a path between them. The travel time from station i to station j through path k is the sum of the travel times of all road segments that make up k. Thus, $TT_{ij} = \sum_{mn \in RS_k} TT_{mn}$ holds, where RS_k is the set of road segments of the path k.

4.1. Calculation of K shortest paths

For each pair of stations and each time interval, we calculate K loopless shortest paths with the Yen algorithm (Yen, 1971), while a modified Dijkstra algorithm (see algorithm 1) with time-dependent travel speed is applied to find the shortest path. The K shortest paths are calculated offline and form a pool of candidates. Thus, when a bus is travelling, the *reselect* operator is applied to recalculate each candidate path. In this way, we avoid using the Dijkstra algorithm online, which results in less computational effort, but at the potential risk of missing the shortest overall. Experiments concerning the influence of the value of K will be presented in the section 6.4.2.

Algorithm 1: Modified Dijkstra's algorithm

Result: find the shortest path from stop A to B
Label A as [0, -], where $[V_A, n_A^{pre}]$ denote the travel time from A and the predecessor of node A;
Label all other nodes in N^g as $[\infty$, -], where N^g is the set of intersections within the rectangular area calculated by equation 1;
Set A as "scanned";
while B *is not scanned* **do**
  Denote the last scanned node n';
  for $\forall n \in N^g$ **do**
    if $\exists$ *road segment* $n'n$ **then**
      Compute $TT_n = t^a_{n'} + TT_{n'n}$, where $TT_{n'n}$ is the travel time from n' to n if departs at time $t^a_{n'}$, TT_n is the travel time from A to n;
      if $TT_n < V_n$ **then**
        relabel n with $[TT_n, n']$;
      end
    end
  end
  Find the unscanned node n with smallest V_n and scan it;
end

When calculating the K shortest paths from A to B, we simply set the Td_{bA} equal to the starting time of the corresponding time interval. Each time interval lasts exactly one hour. For example, to calculate the shortest paths in [7:00,8:00), bus b leaves at 7:00 from A.

However, as seen in the subsection 4, when the departure time is close to the next interval, the travel time can change significantly, and another set of K shortest paths can be obtained. In this case, it seems

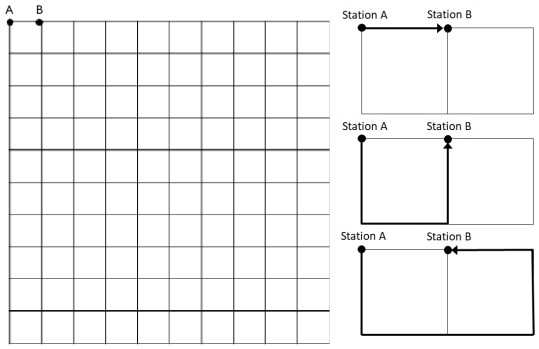


Figure 2: Detours are allowed in calculation of K shortest paths

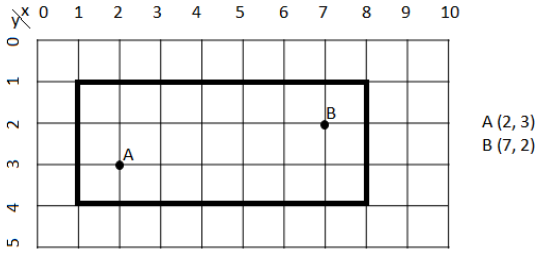


Figure 3: boundaries of detours

important to be more precise and to shorten the time intervals. However, experiments conducted by Huang et al. (2017) in a similar situation conclude that interval times of 30 and 60 minutes lead to the same solution and interval times of 10 minutes lead to a solution only 0.07% better. For this reason, the interval duration will be set at 60 minutes in this paper. Detours between the origin and destination are allowed (shown in figure 2) but are limited in length. For example, in figure 3, a bus moves from station A to B, having coordinates $(x_A, y_A), (x_B, y_B)$ and any detour cannot exceed the rectangular area delimited by

$$\begin{cases} max(min(x_A, x_B) - 1, x_{min}) \\ min(max(x_A, x_B) + 1, x_{max}) \\ max(min(y_A, y_B) - 1, y_{min}) \\ min(max(y_A, y_B) + 1, y_{max}) \end{cases} \tag{1}$$

Where x_{min}, y_{min}, x_{max} and y_{max} are the map boundaries. This is inspired by the situation where bypassing a congested road can reduce travel time, and usually the detour is not much longer than the original in terms of geographic distance. Finally, to simplify the solution and avoid U-turns, we assume that a moving bus cannot take a road segment multiple times between two stations.

4.2. Network-consistent congestion

In most cases, congestion occurs in areas with a dense road network or a large population and is therefore more common in metropolitan areas. Furthermore, traffic jams are usually not restricted to a single road, but extend to the whole network and affect the roads near the start of the congestion.

To generate temporally and geographically correlated travel times, we use the congestion circle model proposed in Lecluyse et al. (2013). The main idea is that a congestion circle starts at a specific place and time and gradually expands (affecting the surrounding road network as well) until it reaches its maximum radius and stagnates. After a certain time, it decreases until the average speed of each road segment is restored. A congestion circle is therefore defined by its location and time of appearance, its maximum radius, and its duration of expansion, stagnation, and shrinkage.

In order to effectively simulate network-compliant traffic congestion, the travel time on each road segment is positively correlated to the percentage of the road segment covered by the congestion circle. Calculating travel times on each segment thus happens on the network layer underlying the graph that connects bus stations, mentioned in section 3.

Equation 2 gives the travel time from i to j when the path of length d_{ij} has a congested road portion of length cls_γ and the impact of congestion on travel time is defined by δ_{ij}. The higher the value of δ_{ij}, the longer the travel time on that portion of the road. In this paper, δ_{ij} is set to 1, so a fully congested road

will take twice as long to travel.

$$TT_{ij}(t) = \overline{TT_{ij}}(t)\left(1 + \delta_{ij}\frac{cls_{\gamma}(t)}{d_{ij}}\right) \tag{2}$$

Calculating the exact time a vehicle reaches and leaves a congested area is very time consuming due to its interdependence with the time-varying radius. We consequently approximate the radius with its static value when a vehicle starts to run on the given segment. This approximation is not necessarily accurate, but we assume that the time it takes to pass each road segment is still much shorter than the time it takes for the congestion circle to change dramatically even in severe congestion.

The main objective of adopting the congestion circle is to generate a reasonable and consistent stochastic deviation with the network from the average value of each time interval and road segment. For more details, we refer to Lecluyse et al. (2013).

5. Solution method

This section first explains how the initial solution is created, taking into account the predicted time-dependent travel time (see subsection 5.1) and then describes in subsection 5.2 how this solution is modified, reacting to the information received in real time.

5.1. Initial solution

This first step determines the passenger-bus assignment and for each bus its route and schedule, using the historical average speeds of each road segment for each hour.

A draft initial solution is first generated by a greedy insertion procedure, described in algorithm 2. Each time a request is inserted in a route, the arrival and departure times in the subsequently scheduled stations are updated using algorithm 3. The initial solution is then improved using variable neighborhood search (VNS), exploring increasingly distant neighborhoods in sequence (Mladenović and Hansen, 1997). Details of this approach are described in algorithm 4. Our input data is a list of requests, each request containing earliest departure time, latest arrival time, and potential departure and arrival stations, as explained in section 3. During the sequential insertion of each request, our objective is to position the requests' pick-up and drop-off stations to minimize the increase in total URT.

In the following algorithms, a solution is considered "feasible" if all the constraints are respected.

Algorithm 2: Build initial solution step 1, greedy insertion

```
Result: Insert each request at the position with least increase of total URT
for each request r do
    for each bus b do
        for each position K do
            for each alternative get-on stop M do
                Temporarily insert M in K;
                Recalculate_arrival_and_departure_time(b, K);
                for each position G > K do
                    for each alternative get-off stop N do
                        Temporarily insert N in G;
                        Recalculate_arrival_and_departure_time(b, G);
                        if feasible and has smaller increase in the sum of URT then
                            Save this solution;
                        end
                    end
                end
            end
        end
    end
end
```

During the greedy phase (see Algorithm 2), an implicit check is performed with the calculation of the maximum arrival time available without violating any time window, also called "Forward Time Slack" (FTS) and introduced by Savelsbergh (1992). The more specific use of this method in ODBRP is called "buffer" in Melis and Sörensen (2021).

Algorithm 3: Recalculate_arrival_and_departure_time(b, K)

```
Result: Obtain bus b's arrival and departure time at each stop L ≥ K
for L ≥ K do
    Ta_bL = Td_b(L−1) + TT_(L−1)L(Td_b(L−1)), calculated by algorithm 1;
    Td_bL = max(max(e_Pb), Ta_bL), where max(e_Pb) is the biggest value of earliest departure among
      all passengers getting on at K;
end
```

The proposed VNS uses three local search operators, *alternative*, *swap* and *reinsert*.
The operator *alternative* (algorithm 5) tries to replace a passenger's drop-off stations with one of its alternative stations (if the passenger has more than one possible drop-off stations within walking distance).If the currently used station is also occupied by other passengers, it will remain but the alternative one will be

inserted just before. Otherwise, it will be safely replaced by the other station.

Algorithm 4: Build initial solution step 2, VNS

```
Result: improve initial solution with VNS
for Each passenger p do
    Alternative();
    Implement the solution with minimal total URT;
end
for Each passenger p do
    Swap();
    Implement the solution with minimal total URT;
end
for Each passenger p do
    Reinsert();
    Implement the solution with minimal total URT;
end
```

Algorithm 5: Alternative()

```
Result: Choose the get-off stop with smaller objective value
Set p_off: p's current get-off stop;
Set p_off': p's alternative get-off stop;
if ∃p_off' then
    p_off = p_off';
    if feasible and smaller objective value then
        Save this solution;
    end
end
```

The operator *swap* (algorithm 6) tries to reduce the total URT by swapping a passenger's get-off station with a previous station. If this station is also occupied by other passengers, we must ensure that the precedence constraint still applies to them.

Algorithm 6: Swap()

```
Set g_on: id of p's get-on stop;
Set g_off: id of p's get-off stop;
for Each position pos < g_off and pos > g_on do
    Swap pos with g_off;
    if feasible and smaller objective value then
        Save this solution;
    end
end
```

Finally, the operator *reinsert* (algorithm 7) is applied to each accepted request, which tries to remove a passenger from a route and reinserted into another if it reduces the total URT. A VNS and these operators were chosen to use similar algorithms for the initial solution and in the real-time step to be able to compare

and understand the efficiency of the real-time actions.

Algorithm 7: Reinsert()

```
for Each bus do
    for each position K do
        for each alternative get-on stop M do
            Temporarily insert M in K;
            Recalculate_arrival_and_departure_time(b, K);
            for each position G > K do
                for each alternative get-off stop N do
                    Temporarily insert N in G;
                    Recalculate_arrival_and_departure_time(b, G);
                    if feasible and smaller objective value then
                        Save this solution;
                    end
                end
            end
        end
    end
end
```

5.2. Real-time Control

As described previously, we assume that the actual travel times are only updated on the road segments traveled between the current arrival station s and the next visited station in the route $s+1$. Travel times further in the route are not updated.

To simulate the real-time update each time a bus reaches a station, the algorithm indexes in chronological order which bus goes to which station and when, as shown in figure 4.

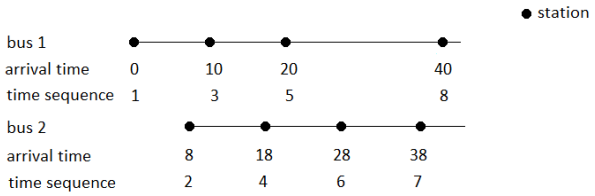


Figure 4: Time sequence of the simulation

Thus, the algorithm knows which bus will be the next to reach its station and its travelling speed to its next station. This idea is summarized in figure 5 in which a bus reaches station s after traveling from $s-1$. The travel speed between stations s and $s+1$ is now revealed, but the travel speed from $s+1$ to $s+2$ is still assumed to be the average statistical speed on this trip for the corresponding time period. As arrival times are also updated, we can compare them with each passenger's latest allowed arrival time, and missed time windows will trigger a modified VNS to improve the quality of the solution.

The VNS proposed for the real-time stage (described in algorithm 8) consists of four operators: *reselect*, *alternative*, *swap* and *reinsert*. The operator *reselect* (algorithm 10) exchanges the previously chosen path from s to $s+1$ by the shortest path from the pre-selected K described in subsection 4.1, with the current travel speed revealed. Thus, this bus will still travel from s to $s+1$, but eventually with a different path.

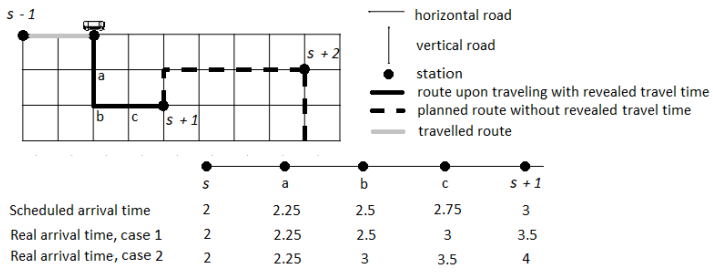


Figure 5: Update travel time at $s+1$

Algorithm 8: Real-time Heuristic

Result: En route adjustment with updated traffic information
while *At least one bus is still running* **do**
  For all buses, travel time from the current position to the next stop are revealed;
  Update_bus_arrival/departure_time(), Algorithm 9;
  Find the next bus b upon arriving at a stop S, set $T_{cur} = Ta_{bS}$, where T_{cur} is the current time;
  Calculate how many passengers on b will be delayed;
  if *delayed passenger(s)* **then**
    Reselect();
    Implement the solution with minimal total URT;
  end
  for *each remaining delayed passenger* P **do**
    Alternative(), Algorithm 5;
    Implement the solution with minimal total URT;
  end
  for *each remaining delayed passenger* P **do**
    Swap(), Algorithm 6;
    Implement the solution with minimal total URT;
  end
  for *each remaining delayed passenger* P **do**
    Reinsert(), Algorithm 7;
    Implement the solution with minimal total URT;
  end
end

Algorithm 9: Update_bus_arrival/departure_ time()

for *each bus* b **do**
  Set *pos*: b's current position;
  Set $S+1$: b's next stop;
  Set T_{cur}: current time;
  Set $TT^{true}_{pos,S+1}$: newly revealed true travel time;
  $Ta_{b,S+1} = T_{cur} + TT^{true}_{pos,S+1}$;
  $Td_{b,S+1} = max(max(e_{Pb}), Ta_{b,S+1})$, where $max(e_{Pb})$ is the biggest value of earliest departure among all passengers getting on at K;
  Recalculate_arrival_and_departure_time$(b, S+2)$;
end

Algorithm 10: Reselect()

Result: Choose the shortest path from S to $S+1$ under current traffic condition
Upon a bus's arrival at stop S:
Set: Smallest arrival time at stop $S+1$ $A^* = \infty$;
for *K candidates of the shortest path S to $S+1$* **do**
  Calculate true arrival time A_{new} at $S+1$;
  if $A_{new} < A^*$ **then**
    This candidate becomes the designated path;
  end
end

Apart from the *reselect* operator, the same local search operators used to generate the initial solution are also applied sequentially in the real-time phase: *alternative*, *swap* and *reinsert*. The main idea of these operators is to adapt the solution to the new speed information obtained.

Although the same three operators are applied in both stages, there are two differences in the application of the operators in real-time control compared to the initial stage. First, the range of application of each operator is limited, in general to a single current route of which the times can be modified. Moreover, *reinsert* can only be applied to passengers not yet aboard. The other difference is with regard to the time window constraints. Specifically, in the initial stage, each operator is allowed to be executed only if the consequent solution still abides by the time window constraints. However, in the real-time stage, it is impossible to guarantee to respect the time windows. Accordingly, violations of time windows can occur when applying the local search operators, on the condition that the total tardiness can be reduced.

Local search operators are performed sequentially and greedily. *reselect* is performed first, followed by the calculation of the number of remaining delayed passengers. Then, the additional operators will try to reduce the URT for each of these passengers. Each operator is executed in a *steepest descent* manner (i.e., the best available solution is chosen), except for the *alternative*, which limits the number of arrival stations studied to two per passenger. Another possibility is to study other pick-up stations for passengers who are not yet on board, but as it is not directly related to the missed time window at the pick-up station, we exclude it in this paper. Operators can also be combined, for example, *reselect* can be performed within the *alternative*, *swap* and *reinsert* operators, if the next station is changed. Nevertheless, only a sequential use of the operators is studied in this paper.

Because of the appointments made with passengers and drivers, it is preferable that these route plans change as little as possible in real time. This is why the operators are called by degree of modification applied to the solution. In particular, the *reselect* will only change the path used, not the order in which stations are visited. Arrival stations may change with the *alternative* operator, but since it is also within walking distance of the associated passenger, it does not make much difference. The operator *swap* modifies the route more significantly since the sequence of stations will be different, and therefore also the routes from one station to another. Finally, unlike other operators, *reinsert* will also involve another vehicle and a driver, and the corresponding passenger will be notified to wait for another vehicle, usually with a different pickup time.

6. Numerical analysis

This section first describes the tested instances and the experimental environment in the subsection 6.3, then the numerical results and analysis are presented in 6.4.

6.1. Map generation

A grid-structured urban area of 6 km × 6 km, divided into 100 blocks of 0.36 square kilometers is created. stations are created at the intersections of the 11 horizontal and vertical streets, so there can be a maximum of 121 stations, as shown in figure 6.

In order to represent the actual urban traffic situation as realistically as possible, we extract the travel speed from Uber Movement (`https://movement.uber.com`), which provides historical data on the average hourly speed of road segments, for several world metropolises. Given the available data and the configuration of the city, data based on the city of San Francisco in January 2020 was chosen.

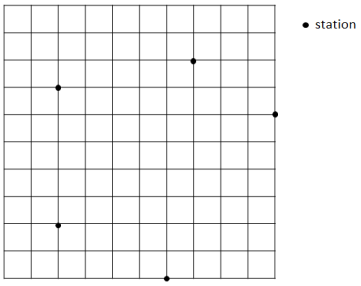


Figure 6: Grid urban map with stations

6.2. Algorithm quality of initial stage

In this paper, we deal with instances too large to be solved with exact methods, even for the initial solution. Instead, we compare our heuristic algorithm with LocalSolver, which is a global optimisation solver, combining exact and heuristic techniques. We limited its execution time to 24 hours (86400 seconds). We compared four different instances' size, with 200, 300, 400 and 600 passengers. The number of buses is fixed to 50, each with capacity equal to 8.

All code was written in C++ and all tests were performed on a computer with an Intel® Core^{TM} i7-8850H 2.60Ghz processor and 16GB RAM running Windows 10.

The results are summarised in table 1. This table shows the average CPU time per instance size to obtain the first feasible solution and the best solution found for each method tested, as well as the URT associated with these solutions. In addition, to clarify the difference between each method at each stage, the GAP between the URT of the solutions returned by the proposed methods and LocalSolver is also displayed.

The first solutions are found in a few seconds by the proposed heuristic and take up to 26 minutes to be obtained by LocalSolver. For both methods, the difference between the objective value of the first solutions and the best known solution is over 40%. Thus, improvement of these solutions using local searches is necessary.

With respect to the best solutions found by each algorithm, LocalSolver always finds better solutions than the proposed method. However, LocalSolver needs several hours to converge for instances with 300 or more queries while the proposed heuristic needs only a few minutes to find solutions of reasonable quality.

For the instance size with 600 passengers, LocalSolver was unable to find a feasible solution within 24 hours.

The proposed method therefore provides a trade-off between the execution time and the URT of the corresponding solutions.

Despite the fact that our method is fully greedy, in this section we explicitly relax the algorithm 4 for the purpose of testing it 6.2. More specifically, we allow it to get rid of the local optima and have a longer execution time, with the details explained in the next paragraph. The purpose of this experiment is simply to test the performance of the algorithm, if it is allowed . However, we abide again by the fully greedy version for the initial solutions of the instances in the following subsection 6.4, in order to have speedy solutions.

The relaxed version of algorithm 4 functions as follows. We define a "round" as executing 4 once. If no better solution than the current best one is found in the last round, we allow the algorithm to randomly accept a candidate solution that is not more than 5% worse compared with the current best solution, which becomes the new incumbent solution. The rounds continue until the CPU time reaches the bound, which is set to 10 minutes for 200 instances, while 25 minutes for 300 and 400. Therefore, the results are obtained and listed in table 2. As shown, the relaxed proposed algorithm performs drastically better: in particular,

equal values are obtained within 6 minutes for the 200 size; smaller gaps are achieved for the 300 and 400 sizes, on average after approximately 13 and 16 minutes respectively. In summary, this proves the reliability of the proposed method.

Instance size	First solution					Best solution				
	URT			CPU time (s)		URT			CPU time (s)	
	LocalSolver	Heuristic	GAP (%)	LocalSolver	Heuristic	LocalSolver	Heuristic	GAP (%)	LocalSolver	Heuristic
200	2453	**2321**	-5.4	32	**4**	**1299**	1343	3.4	1236	**71**
300	2931	**2750**	-6.2	209	**11**	**1638**	1782	8.8	34939	**132**
400	3153	**2870**	-9.0	1570	**14**	**1934**	1984	2.6	47051	**193**

Table 1: Comparison between the proposed method and LocalSolver to generate the initial solution

Instance size	Best solution				
	URT			CPU time (s)	
	LocalSolver	Heuristic	GAP (%)	LocalSolver	Heuristic
200	**1299**	**1299**	0	1236	**354**
300	**1638**	1679	2.5	34939	**788**
400	**1934**	1969	1.8	47051	**967**

Table 2: Comparison between the relaxed proposed method and LocalSolver to generate the initial solution

6.3. Tested instances

To test the full algorithm, including the procedure for the real-time modifications, a set of 40 instances was generated, each composed of 794 to 804 passengers (800 on average), and a fleet of homogeneous minibuses with a capacity of 8 is used. In terms of the fleet size, we have four different levels: 50, 100, 150 and 200. Passengers' requests are spread over 50 stations (each located at an intersection), and they travel during the morning rush hour from 7:00 am to 10:00 am, divided into 3 one-hour time slots.

The objective here is to minimise the total delay, measured as the total tardiness. Indeed, we have considered that reducing the total tardiness will also lead to a reduction of the number of delayed passengers and to an improvement of the total URT. The reduction of the number of delayed passengers and the minimisation of the total URT are consequently not explicitly taken into account in our objective function.

6.4. Numerical results and analysis

This subsection is organized as follows. First, the impact of congestion on URT is measured in 6.4.1. Subsequently, the effectiveness with regard to reduced tardiness of the LS operators is investigated. More specifically, the *reselect* operator together with variant K is illustrated in 6.4.2. Furthermore, the overall effectiveness of the VNS algorithm is analysed in 6.4.3.

6.4.1. Effect of congestion on the URT

We first experiment how the URT increases with different levels of congestion. For this purpose, an indicator on the degree of congestion called "average speed", which is the ratio between the main current speeds and the main theoretical speeds, is calculated (equation 3). Thus, a lower average speed is equivalent to a higher level of congestion.

$$Average\ speed = \frac{\sum_{b}^{B} \sum_{i}^{SS_b} TT_{i,i+1}^{true}}{\sum_{b}^{B} \sum_{i}^{SS_b} TT_{i,i+1}^{theory}} \tag{3}$$

Figure 7 shows the increase of URT in function of the average speed. As expected, the more the average speed is impacted, the higher the URT.

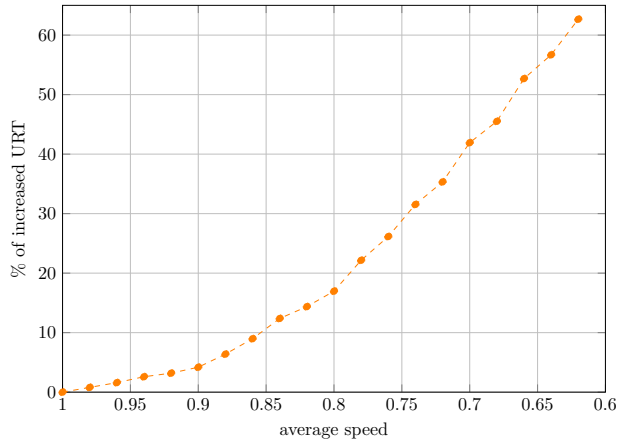


Figure 7: Increase in total URT with decreasing average speed

In the equation 2, the weighting factor (δ) has been empirically set at 1, so in the worst case, the average speed will be 0.5. In this situation, passengers' travel time will be doubled for any road segment, leading to an equivalent graph topology. It also makes it impossible to carry a passenger in time, as the time windows are only 60% longer than the shortest non-congested route (as explained is section 3). The average speed was therefore scaled in the range [0.62, 0.98] in order to have a real-time traffic network distinct from the theoretical case.

6.4.2. Number of shortest path candidates

As described in section 4.1, an offline candidate pool of K shortest paths is calculated for each pair of stations and each time interval. Thus, in the online phase, the K shortest paths are updated with the real travel time in order to use the shortest path between stations s and $s+1$. So, the higher K is, the more alternative paths are available. Calculating the shortest path online from zero requires time and a high value of K could lead to finding the shortest path in a longer computation time.

Experiments were made from 40 instances with 50 buses in order to determinate the best value of K. The efficiency measured by the percentage reduction in the total amount of tardiness of all delayed passengers) of different K were compared (see figure 8).

The graph reveals that alternative paths can effectively reduce the total delay time of passengers. In particular, this operator is very useful when congestion is low, with a reduction of up to nearly 40% when K is strictly greater than 2. The utility of this operator decreases as congestion increases and becomes negligible when the average speed is below 0.7. Indeed, when congestion is low, the congestion circle is small, and some of the alternative paths are able to avoid the circle. As congestion increases, more areas are affected, making it more difficult to find an alternative path outside the circle. At the same time, the greater the congestion, the greater the number of delayed passengers. Thus, the reduction in travel time of a delayed passenger contributes less proportionally to the reduction in total travel time of the user.

Furthermore, in the case of medium congestion, the topology is more affected and the actual shortest path among the K-candidates is more likely to change. While when the congestion is really severe, according to our congestion circle model, everywhere tends to become more and more equally congested, which means the topology of the travel time graph again becomes very similar to the one used to create the initial solution, thus the effectiveness is lower.

The study of the different values of K indicates that a value of $K = 3$ reduces 61% more the total tardiness than a value of $K = 2$. However, the gain in total URT to have an higher value of K is negligible and increases the total execution by more than 30%. Thus, we consider a value of $K = 3$ for the rest of the paper.

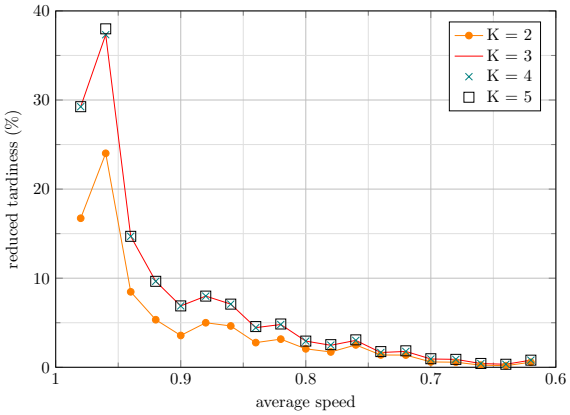


Figure 8: Improved total tardiness, K shortest paths

Due to the FIFO property, the *reselect* operator will never deteriorate the solution, but it only applies until the next station, then another update can be necessary. When K = 1, no alternative path is used, so the total delayed time decrease is 0%.

The experiments were performed with 40 instances each having 50 buses to fix this parameter. A more extensive investigation of flexibility and the impact of the number of buses could be interesting for future work.

6.4.3. Effectiveness of local search operators

As described in subsection 5.2, in the real-time control stage, three more operators are applied to improve the solution quality. The efficiency of the four operators as a whole (i.e. the algorithm 8), is measured by the tardiness and the percentage of late passengers, and the results are respectively in figures 9 and 10.

The average URT is around 5 minutes without congestion, and 7-8 minutes with high congestion (more details are available in B.4 in Appendix B). The average execution time required to adapt a route each time a bus reaches a station varies from 10 ms when the average speed is equal to 0.98, to 227 ms when the average speed is equal to 0.62. The algorithm is therefore fast enough to be used in real-time.

As shown in figures 9 and 10, the solver is effective to reduce the delay of late passengers, especially when the congestion level is low or moderate. We further studied the impact of flexibility by testing the same 40 instances with 100, 150, 200 vehicles. Similarly, the exact delays and URT are given in the tables B.5, B.6 and B.7 in the Annex.

With 100 buses, the results are in a similar trend as 50, but the average tardiness is maintained below 7 minutes, and the maximum tardiness reached by a passenger is 31 minutes. With 150 buses, the average tardiness is further improved to be less than 5 minutes, and the maximum tardiness is 27 min. Finally, with 200 buses, where each bus carries only 4 passengers on average, the tardiness and late passengers after a local search are illustrated by figures 11 and 12 respectively.

As we expected, minimizing total tardiness implicitly reduces the number of late passengers and the maximum tardiness (see tables B.4-B.7), at least when the solver has a lot of flexibility (i.e. when the number of buses is large and/or when the congestion is low). However, when the average speed is less than 0.65 and only 50 buses are available, the solver may sacrifice some passengers in order to reduce the delay of other passengers.

One other reason which contributes to delay passenger is the restrictions made in the local search to not impact the quality of service. Indeed, if a passenger will soon be picked up, the solver cannot change the passenger's affected bus or station, even if it will contribute to reduce the tardiness. For the same reason, the next station planned for a route will not be changed. Theses constraints represent the trade-off between the flexibility of the solver and the capacity of adaption needed from the customers and the drivers.

Another possible reason could be the lack of information on future traffic. In order to reduce the execution time, we consider that the solver only knows the actual speed of the edges being traveled. Frequent updating

of the entire network can be envisaged as part of future work.

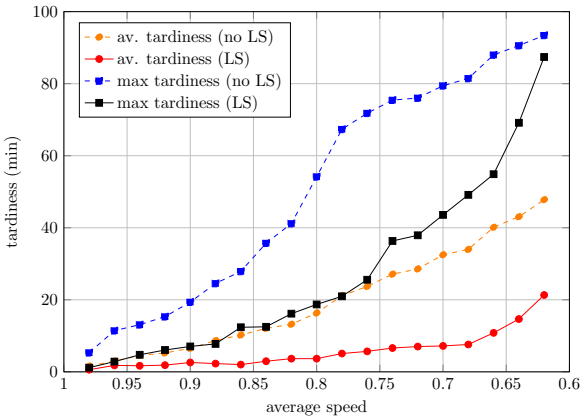


Figure 9: Average and maximum tardiness, 50 buses

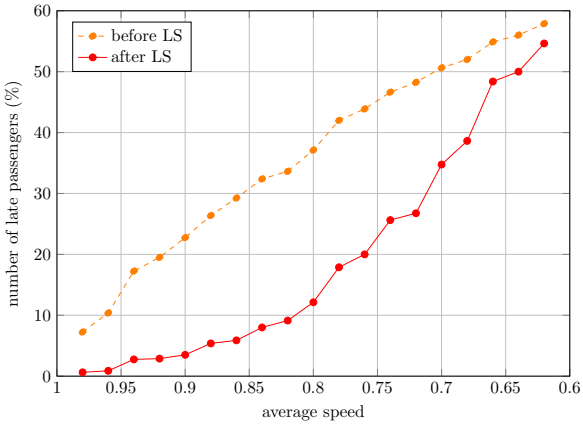


Figure 10: Proportion of late passengers, 50 buses

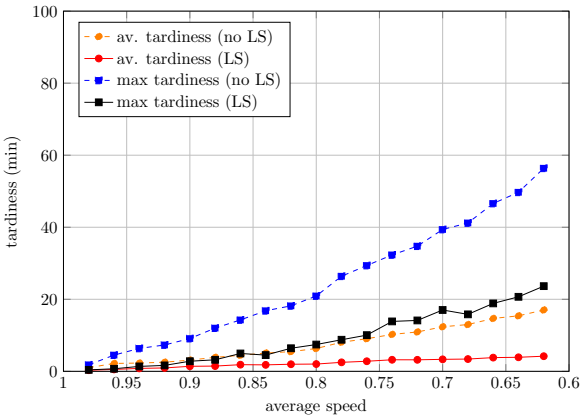


Figure 11: Average and maximum tardiness, 200 buses

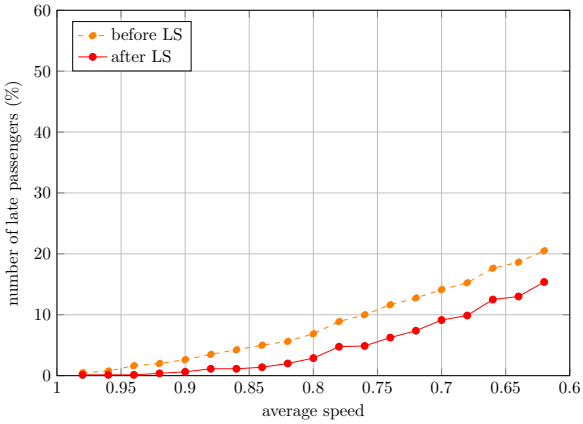


Figure 12: Proportion of late passengers, 200 buses

7. Conclusions and future work

In this paper, we have considered a variant of the on-demand bus routing problem that considers real-time traffic information. The corresponding problem is solved by VNS that uses four operators: *alternative*, *swap* and *reinsert*, as well as *reselect* as an integrated decision. The model is formulated and simulated under both deterministic and dynamic traffic information. The first part concerns time-dependent travel speed and its stochastic variant is generated using a congestion circle consistent with the network. The solver has been tested on 40 instances and the experimental results demonstrate the benefits of real-time control operators under certain conditions.

Experiments show how choosing the shortest path among the 3 considered candidates always leads to better or equivalent solutions to the original due to the FIFO property. However, this efficiency decreases with congestion. It is therefore unnecessary to calculate the shortest path from scratch in real-time. The results also indicate that a larger number of candidate paths will increase the computation time without improving the quality of the solution for instances with a small number of buses. Further experiments should be considered to determine whether this assumption is still valid with a larger number of buses.

Furthermore, the VNS algorithm is demonstrated to reduce the average delay of the passengers. Thus, when congestion is reasonable, this algorithm is able to reduce in real-time its impact on the total tardiness and implicitly the number of delayed passengers through a modification of the initial solution.

Future work may be conducted by using real data to test the solver on real congestion and on instances with different topologies. The adaptation of this solver with instances combining ODBRP with multiple variations such as stochastic travel time or stochastic demand can lead to a new framework able to solve more complex and realistic instances.

Acknowledgements

This research was supported by a grant of the Flemish Fund for Scientific Research (FWO Strategic Basic Research Fellowship grant).

CRediT authorship contribution statement

Ying Lian: Conceptualization, Methodology, Software, Formal analysis, Investigation, Drafting, Writing - review and editing, Visualization. **Flavien Lucas**: Methodology, Validation, Drafting, Writing - review and editing, Formal analysis, Visualization, Supervision. **Kenneth Sörensen**: Conceptualization, Validation, Resources, Writing - review and editing, Project administration, Funding acquisition, Supervision.

References

Y. G. Anil Rao, N. S. Kumar, H. S. Amaresh, and H. V. Chirag. Real-time speed estimation of vehicles from uncalibrated view-independent traffic cameras. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–6, 2015.

C. E. Cortés, A. Núñez, and D. Sáez. Hybrid adaptive predictive control for a dynamic pickup and delivery problem including traffic congestion. *Adaptive Control and Signal Processing*, 22:103–123, 2008.

F. Dehne, M. Omran, and J. Sack. Shortest paths in time-dependent fifo networks. *Algorithmica*, 62 (1–2): 416–435, 2012.

R. Eglese, W. Maden, and A. Slater. A road timetable to aid vehicle routing and scheduling. *Computers & Operations Research*, 33(12):3508 – 3519, 2006. Part Special Issue: Recent Algorithmic Advances for Arc Routing Problems.

B. Fleischmann, M. Gietz, and S. Gnutzmann. Time-varying travel times in vehicle routing. *Transportation Science*, 38(2):160–173, 2004a.

B. Fleischmann, S. Gnutzmann, and E. Sandvoss. Dynamic vehicle routing based on online traffic information. *Transportation Science*, 38(4):420–433, 2004b.

L. Fu. Improving paratransit scheduling by accounting for dynamic and stochastic variations in travel time. *Transportation Research Record*, 1666(1):74–81, 1999.

L. Fu. Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation Research Part B: Methodological*, 36(6):485 – 506, 2002.

T. Garaix, C. Artigues, D. Feillet, and D. Josselin. Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research*, 204(1):62 – 75, 2010.

M. Gendreau, G. Ghiani, and E. Guerriero. Time-dependent routing problems: A review. *Computers & Operations Research*, 64:189 – 197, 2015.

A. V. Hill and W. C. Benton. Modelling intra-city time-dependent travel speeds for vehicle scheduling problems. *Journal of the Operational Research Society*, 43(4):343–351, 1992.

S. C. Ho, W. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering, and T. W. Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395 – 421, 2018.

S. G. Ho, H. W. Koh, R. R. Pandi, S. C. Nagavarapu, and J. Dauwels. Solving time-dependent dial-a-ride problem using greedy ant colony optimization. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 778–785, 2018.

Y. Huang, L. Zhao, T. Van Woensel, and J.-P. Gross. Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological*, 95:169 – 195, 2017.

O. Ibarra-Rojas, F. Delgado, R. Giesen, and J. Muñoz. Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological*, 77:38 – 75, 2015.

S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379 – 396, 2003.

D. E. Kaufman and R. L. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *I V H S Journal*, 1(1):1–11, 1993.

G. Kim, Y. S. Ong, T. Cheong, and P. S. Tan. Solving the dynamic vehicle routing problem under traffic congestion. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2367–2380, 2016.

A. Kok, E. Hans, and J. Schutten. Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, 39(5):910 – 918, 2012.

C. Lecluyse, K. Sörensen, and H. Peremans. A network-consistent time-dependent travel time layer for routing optimization problems. *European Journal of Operational Research*, 226(3):395 – 413, 2013.

W. Maden, R. Eglese, and D. Black. Vehicle routing and scheduling with time-varying data: A case study. *Journal of the Operational Research Society*, 61 (3):515–522, 2010.

C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26:185–200, 1992.

C. Malandraki and R. B. Dial. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research*, 90(1):45 – 55, 1996.

L. Melis and K. Sörensen. The static on-demand bus routing problem: large neighborhood search for a dial-a-ride problem with bus station assignment. *International Transactions on Operations Research*, forthcoming, 2021.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11): 1097 – 1100, 1997.

N. Rincon-Garcia, B. Waterson, and T. Cherrett. A hybrid metaheuristic for the time-dependent vehicle routing problem with hard time windows. *International Journal of Industrial Engineering Computations*, 8(1):141–160, 2017.

M. W. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA J. Comput.*, 4 (2):146–154, 1992.

M. Schilde, K. Doerner, and R. Hartl. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1):18 – 30, 2014.

M. Setak, M. Habibi, H. Karimi, and M. Abedzadeh. A time-dependent vehicle routing problem in multigraph with fifo property. *Journal of Manufacturing Systems*, 35:37 – 45, 2015.

R. Spliet, S. Dabia, and T. V. Woensel. The time window assignment vehicle routing problem with time-dependent travel times. *Transportation Science*, 52(2):261–276, 2018.

D. Taş, N. Dellaert, T. van Woensel, and T. de Kok. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, 40(1):214 – 224, 2013.

D. Taş, M. Gendreau, N. Dellaert, T. van Woensel, and A. de Kok. Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3):789 – 799, 2014. Vehicle Routing and Distribution Logistics.

T. Vidal, R. Martinelli, T. A. Pham, and M. H. Hà. Arc routing with time-dependent travel times and paths. *Transportation Science*, 2021.

J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.

Appendix A. Mathematical model

Table A.3: Variables and parameters of the ODBRP

x_{snb}	1 if the n-th station of bus b is bus station s and 0 otherwise
y^u_{pnb}	1 if passenger p is picked up at the n-th station of bus b and 0 otherwise
y^o_{pnb}	1 if passenger p is dropped off at the n-th station of bus b and 0 otherwise
q_{nb}	net number of passengers picked up (or dropped off) at the n-th station of bus b
t^a_{nb}	arrival time of bus b at its n-th station
t^d_{nb}	departure time of bus b at its n-th station
T_p	user ride time of passenger p
B	the fleet of buses
P	the set of transportation requests, $\|P\|$ denotes the number of requests
S	the set of bus stations
Q	capacity of bus
a^u_{ps}	1 if passenger p can be assigned to station s for pick-up
a^o_{ps}	1 if passenger p can be assigned to station s for drop-off
e_p	earliest pick-up time for passenger p
l_p	latest drop-off time for passenger p
$TT_{ss'}(t)$	travel time between station s and station s' if departs at time t

$$\min \text{URT} = \sum\nolimits_p T_p \tag{A.1}$$

s.t.

$$\sum\nolimits_s x_{snb} \leq 1 \qquad \forall n \in N, b \in B \tag{A.2}$$

$$\sum\nolimits_s (x_{snb} - x_{s(n+1)b}) \geq 0 \qquad \forall n \in N, b \in B \tag{A.3}$$

$$M \sum\nolimits_s x_{snb} - \sum\nolimits_p (y^u_{pnb} + y^o_{pnb}) \geq 0 \qquad \forall n \in N, b \in B \tag{A.4}$$

$$\sum\nolimits_s x_{snb} - \sum\nolimits_p (y^u_{pnb} + y^o_{pnb}) \leq 0 \qquad \forall n \in N, b \in B \tag{A.5}$$

$$x_{snb} + y^u_{pnb} - a^u_{ps} \leq 1 \qquad \forall s \in S, n \in N, p \in P, b \in B \tag{A.6}$$

$$x_{snb} + y^o_{pnb} - a^o_{ps} \leq 1 \qquad \forall s \in S, n \in N, p \in P, b \in B \tag{A.7}$$

$$t^a_{(n+1)b} - t^d_{nb} - TT_{ss'}(t^d_{nb}) + (x_{snb} + x_{s'(n+1)b} - 2)(-M) \geq 0 \qquad \forall s, s' \in S \mid s \neq s', n \in N, b \in B \tag{A.8}$$

$$t^d_{nb} - e_p + (y^u_{pnb} - 1)(-M) \geq 0 \qquad \forall p \in P, n \in N, b \in B \tag{A.9}$$

$$t^a_{nb} - l_p + (y^o_{pnb} - 1)M \leq 0 \qquad \forall p \in P, n \in N, b \in B \tag{A.10}$$

$$\sum\nolimits_n (n y^u_{pnb} - n y^o_{pnb}) \leq 0 \qquad \forall p \in P, b \in B \tag{A.11}$$

$$\sum\nolimits_n (y^u_{pnb} - y^o_{pnb}) = 0 \qquad \forall p \in P, b \in B \tag{A.12}$$

$$\sum\nolimits_b \sum\nolimits_n y^u_{pnb} \leq 1 \qquad \forall p \in P \tag{A.13}$$

$$x_{snb} + x_{s(n+1)b} \leq 1 \qquad \forall s, n, b \tag{A.14}$$

$$\sum\nolimits_p (y^u_{pnb} - y^o_{pnb}) - q_{nb} = 0 \qquad \forall n \in N, b \in B \tag{A.15}$$

$$\sum\nolimits_{n' \leq n} q_{nb} \leq Q \qquad \forall n, n' \in N \mid n \geq n', b \in B \tag{A.16}$$

$$T_p + (2 - y^o_{pn'b} - y^u_{pnb})M - t^a_{n'b} + t^d_{nb} \geq 0 \qquad \forall n, n' \in N \mid n' > n, p \in P, b \in B \tag{A.17}$$

$$\sum\nolimits_p \sum\nolimits_b \sum\nolimits_n y^u_{pnb} = |P| \tag{A.18}$$

$$x_{snb} \in \{0, 1\} \qquad \forall s \in S, n \in N, b \in B \tag{A.19}$$

$$y^u_{pnb} \in \{0, 1\} \qquad \forall p \in P, n \in N, b \in B \tag{A.20}$$

$$y^o_{pnb} \in \{0, 1\} \qquad \forall p \in P, n \in N, b \in B \tag{A.21}$$

$$q_{nb} \in Z \qquad \forall n \in N, b \in B \tag{A.22}$$

The objective function is to minimize total URT. Constraints (A.2) enforce the fact that a bus can only stop at one station at the same time. Constraints (A.3) make sure stations that the positions used in the bus route are used consecutively and start at the first position. Constraints (A.4) and (A.5) respectively enforce a bus to stop at one station if and only if at least one passenger uses it either to board or alight. Constraints (A.6) and (A.7) respectively impose that a station is designated to a passenger to board/alight only if the station belongs to the passenger, i.e. it is within the predefined walking distance. Constraints (A.8) impose for any two consecutive stations, the arrival time at the later station is (larger than or) equal to the departure time at the previous one plus the travel time, where the travel time depends on the departure time. Constraints (A.9) guarantee the departure time at a passenger's pickup station is greater than or equal to the earliest allowed value. Correspondingly, constraints (A.10) guarantee the arrival time at a passenger's drop-off station is smaller or equal to the latest allowed value. Constraints (A.11) impose that the pickup station precedes the corresponding drop-off one for any passengers. Constraints (A.12) enforce each passenger gets on and gets off the same bus. Constraints (A.13) make sure each passenger is served at most once. Together with constraints (A.18), every request is served once and only once. Constraints (A.14)

forbid two consecutive stations be the same. Constraints (A.15) calculate the net capacity at each station, which is equal to the number of passengers getting on minus the one of getting off. Consequently, constraints (A.16) forbid the violation of bus capacity. Constraints (A.17) calculate the URT of each passenger, i.e. the arrival time at the get-off station minus the departure time at the get-on station. Constraints (A.19 - A.22) define the range for each variable.

Appendix B. Tables of results

Here we present the detailed results of tardiness and URT, for different numbers of buses. In the tables, "av", "pas", "tar", "rid" and "med" corresponds respectively to the average, the passengers, the tardiness, the riding time and the median. Finally, the row with "w" corresponds to results with the use of the solver. Each number corresponds to the average of the results obtained for the 40 instances tested.

Table B.4: Results of tardiness and URT, 50 buses

av speed	0,98	0,94	0,9	0,84	0,8	0,74	0,7	0,62
late pas	7,3%	17,3%	22,8%	32,4%	37,1%	46,6%	50,6%	57,9%
late pas w	0,6%	2,8%	3,5%	8,0%	12,1%	25,6%	34,8%	54,6%
av tar	1,58	4,53	6,54	12,07	16,34	27,12	32,5	47,79
av tar w	0,59	1,69	2,6	2,95	3,66	6,61	7,19	21,33
med tar	1,34	4,39	5,47	11,01	14,98	24,63	30,89	49,64
med tar w	0,54	1,41	1,82	2,24	2,71	4,59	6,05	19,52
max tar	5,32	13,09	19,36	35,72	54,17	75,43	79,43	93,42
max tar w	1,16	4,73	7,05	12,5	18,71	36,34	43,58	87,41
av rid	5,05	5,14	5,22	5,63	5,86	6,59	7,11	8,15
av rid w	5,08	5,17	5,25	5,76	6,05	6,97	7,5	8,62
med rid	4,6	4,92	5,04	5,45	5,59	6,08	6,29	7,33
med rid w	4,84	5,18	5,53	5,84	6,4	7,6	8,2	9,85
max rid	17,6	17,6	17,6	20,09	20,44	25,67	25,67	34,4
max rid w	46,67	63,69	70,38	75,69	95,75	102,07	110,23	113,55

Table B.5: Results of tardiness and URT, 100 buses

av speed	0,98	0,94	0,9	0,84	0,8	0,74	0,7	0,62
late pas	4,0%	11,0%	14,9%	22,0%	25,5%	35,3%	39,4%	47,5%
late pas w	0,4%	1,9%	3,1%	6,0%	9,1%	15,9%	20,6%	39,9%
av tar	1,43	3,44	4,83	8,47	11,26	19,13	23,53	33,6
av tar w	0,71	1,11	1,68	2,54	3,04	4,57	4,93	6,4
med tar	1,2	2,78	4,02	7,26	10,07	17,47	21,79	32,59
med tar w	0,72	0,85	1,38	1,94	2,25	3,56	3,87	5,17
max tar	4,55	13,06	18,07	29,3	36,5	60,05	68,99	85,52
max tar w	1,31	4,03	5,4	10,1	12,24	22,58	24,78	30,66
av rid	4,89	5,11	5,27	5,69	6,03	7,02	7,55	8,81
av rid w	5,07	5,58	5,95	6,86	7,49	9,25	10,11	12,25
med rid	4,35	4,53	4,63	4,89	5,15	5,91	6,41	7,65
med rid w	4,42	4,67	4,87	5,28	5,61	6,7	7,34	9,16
max rid	17	17,98	18,75	20,64	22,06	26,5	28,35	32,84
max rid w	36,8	43,88	46,45	54,67	58,78	69,79	73,36	79,65

Table B.6: Results of tardiness and URT, 150 buses

av speed	0,98	0,94	0,9	0,84	0,8	0,74	0,7	0,62
late pas	1,0%	3,1%	5,1%	10,3%	13,3%	20,4%	24,0%	32,3%
late pas w	0,3%	0,6%	1,6%	3,3%	5,5%	10,6%	15,1%	23,5%
av tar	1,18	2,74	3,7	6,29	8,04	13,45	16,31	22,64
av tar w	0,38	0,84	1,39	2	2,57	3,53	4,14	4,89
med tar	1,07	2,31	3,13	5,27	6,88	11,44	14,31	20,86
med tar w	0,39	0,66	1,23	1,54	2,08	2,73	3,08	3,7
max tar	3,01	8,82	12,15	21,08	27,88	45,48	52,85	68,9
max tar w	0,48	1,89	4,01	7,83	9,33	16,24	19,93	26,89
av rid	4,88	5,08	5,23	5,62	5,93	6,87	7,42	8,71
av rid w	4,94	5,25	5,49	6,15	6,62	7,92	8,8	10,53
med rid	4,35	4,51	4,61	4,88	5,12	5,84	6,28	7,56
med rid w	4,36	4,57	4,71	5,06	5,35	6,2	6,88	8,18
max rid	17,02	17,9	18,66	20,49	21,67	26,23	27,89	32,54
max rid w	25,22	35,98	39,97	51,28	55,6	61	68,7	70,9

Table B.7: Results of tardiness and URT, 200 buses

av speed	0,98	0,94	0,9	0,84	0,8	0,74	0,7	0,62
late pas	0,5%	1,6%	2,6%	5,0%	6,9%	11,6%	14,1%	20,5%
late pas w	0,1%	0,1%	0,6%	1,4%	2,9%	6,3%	9,1%	15,4%
av tar	1,02	2,3	3,11	5,08	6,33	10,25	12,37	17,03
av tar w	0,31	0,82	1,4	1,8	2,02	3,21	3,33	4,2
med tar	1,12	2,02	2,48	4,13	5,3	8,71	10,54	14,88
med tar w	0,41	0,95	1,32	1,63	1,61	2,47	2,56	3,03
max tar	1,81	6,36	9,1	16,82	20,92	32,32	39,38	56,38
max tar w	0,41	1,37	2,81	4,51	7,44	13,86	17,03	23,65
av rid	4,87	5,03	5,16	5,53	5,81	6,69	7,24	8,48
av rid w	4,89	5,1	5,27	5,75	6,11	7,26	7,95	9,56
med rid	4,34	4,5	4,57	4,81	5,06	5,74	6,16	7,35
med rid w	4,35	4,53	4,61	4,93	5,18	5,91	6,46	7,65
max rid	16,99	17,47	17,97	19,61	20,5	24,72	26,73	32,07
max rid w	20,53	26,1	32	39,22	42,75	56,89	61,72	68,93