A column generation algorithm for the demand-responsive feeder service with mandatory and optional, clustered bus-stops

Bryan David Galarza Montenegro
Department of Engineering Management (ENM), University of Antwerp
bryan.galarzamontenegro@uantwerpen.be

Kenneth Sörensen
Department of Engineering Management (ENM), University of Antwerp
kenneth.sorensen@uantwerpen.be

Pieter Vansteenwegen
KU Leuven Mobility Research Centre - CIB, KU Leuven
pieter.vansteenwegen@kuleuven.be

Data availability statement
The authors confirm that the data supporting the findings of this study are available within the article and its supplementary materials.

Funding statement
This project was supported by the FWO (Research Foundation Flanders) project G.0759.19N.

A column generation algorithm for the demand-responsive feeder service with mandatory and optional, clustered bus-stops

Bryan David Galarza Montenegro[a], Kenneth Sörensen[a], Pieter Vansteenwegen[b]

[a]*Department of Engineering Management (ENM), University of Antwerp, Prinsstraat 13, Antwerp, 2000, Antwerp, Belgium*
[b]*KU Leuven Mobility Research Centre - CIB, KU Leuven, Celestijnenlaan 300, Leuven, 3001, Vlaams-Brabant, Belgium*

Abstract

With the rise of smart cities in the near future, it will be possible to collect relevant data from passengers in order to improve the quality of transport services. In this paper, a mathematical model and algorithm are developed to plan the trips of the buses in a demand-responsive feeder service. A feeder service transports passengers from a low-demand area, like a sub-urban area, to a transportation hub, like a city center. The feeder service modeled in this paper considers two sets of bus stops: mandatory stops and optional stops. Mandatory stops are always visited by a bus, while optional stops are only visited when a client nearby makes a request for transportation. Passengers are assigned to a bus stop within walking distance. This in turn, gives the service both flexibility through the changing timetables and routes of the buses and some predictability due to the mandatory stops. To optimize the performance of the service, mathematical modeling techniques to improve the model's runtime are developed. It is concluded that a combination of column generation and the separation of sub-tour elimination constraints decreases the computing time of small and midsize instances significantly.

Keywords: flexible bus services, on-demand transportation, feeder service, demand-responsive transportation, combinatorial optimization, column generation

1. Introduction

Mobility is essential for the growth of a society as it enables accessibility to opportunities, social networks, goods and services. Local public transport contributes greatly to urban mobility. Over 57.6 billion public transport journeys were recorded in the European Union in 2014, where 55.7% of these journeys were road-based services [1]. An adequate public transport service is thus essential and can reduce social exclusion and poverty [2]. Over the past few decades, mobility has increased substantially across many European cities. However, this has also resulted in an increase in congestion and pollution due to the preference for private transport over the collective use of public transport ([3, 4]), underlying the need to increase the attractiveness of public transport.

To become more market-oriented and competitive, the service quality of public transport needs to improve, which can only be achieved through a clear understanding of travel behavior, consumer needs and expectations [5]. Research has shown that reliability is a decisive factor. Rather than waiting time, the uncertainty of when transport will arrive is of most importance. Attributes such as frequency, comfort and arrival time at the destination are also highly valued by consumers. These attributes are key elements of consumer satisfaction ([6, 7]). Other attributes may also have a positive effect on satisfaction and can represent great potential for improvement. For instance, service providers should make clear and simple information available to the public ([8, 5]).

In the future, smart cities may provide opportunities to improve passenger satisfaction with public transport. A smart city is an innovative city that uses information and communication technologies and other means to increase the effectiveness of urban management and services, and consequently improve the living standards of its citizens [9]. In a smart city scenario, it will be possible to collect real-time data concerning the potential passengers, like their requested arrival time and their current location etc. Service providers will also be able to communicate bus arrival times and other information in real-time to their passengers and drivers through web or mobile interfaces as well as through displays on-site. This will create a two-way communication system between the passenger and the service providers, which offers a number of opportunities to improve the latter's service quality.

Traditional Transport Services (TTS) are composed of a set of lines that follow predefined routes and timetables. One crucial advantage of TTS is their low operational costs, arising from their ability to transport large groups of passengers collectively. Furthermore, TTS are efficient in high demand areas as they can sufficiently meet the constant demand of all passengers without requiring routes or timetables to be customized ([10, 11]). The predictable nature of TTS due to fixed timetables makes them highly accessible to most commuters. However, the inflexibility of TTS can also be viewed as a limitation, since it renders TTS inadequate in settings where demand for transportation is sparse and/or constantly changing. Furthermore, traditional bus services are developed using historical, aggregate data and therefore might not cater to the requirements caused by less predictable, ever-changing and more geographically and temporally dispersed travel patterns of citizens today [12]. As such, service providers are unable to effectively handle this demand with their current capabilities and resources. One of the consequences is that passengers utilizing TTS frequently experience long travel times which may result in frustration and client dissatisfaction.

Due to the limitations of TTS, On-Demand Transportation Services (ODTS) are popping up to deal with this volatile demand for transportation. On the one hand, ODTS operate only when there is demand for transportation and are thus better equipped to meet the passenger's expectations [13]. On the other hand, ODTS are often quite expensive and are not applicable on a large scale due the high complexity of scheduling the passenger's requests for a ride and the routing of the vehicles. These services also do not offer a solution to deal with unknown demand for transportation, i.e., passengers that do not request a ride because they are unfamiliar with the service, but could still benefit from such a ride. There seems to be a window of opportunity for public transport services that combine characteristics of both TTS and ODTS.

This paper will focus on developing an exact algorithm for optimizing the performance of a Demand-Responsive Feeder Service (DRFS), introduced in Galarza Montenegro et al. [14]. The DRFS integrates positive characteristics of TTS as well as those of ODTS. A feeder service is defined as a service that transports passengers, from typically sparsely populated areas, to areas with a high demand for transportation, where the passengers can continue their journey. All passengers will thus have the same destination, but different origins. The DRFS can have one or multiple bus lines and each bus line serves two sets of bus-stops: mandatory stops and (clustered) optional stops. The mandatory stops need to be visited by each bus serving a certain line. The optional stops are only visited by a bus when a client, with origin within walking distance of this stop, needs to be picked-up. The service therefore assumes that users can order a ride on the bus line, like e.g., in a taxi. This implies that the buses can have different routes and different timetables according to the demand, while there is still a factor of predictability due to the mandatory stops. The mandatory stops serve as a safety-net for the unknown demand, as potential passengers may simply take the bus at one of the mandatory stops, even without making a formal request for transportation.

In the DRFS, potential passengers make a request for transportation to the transportation hub, by stating their current location and their latest arrival time. We assume that all requests for transportation are known before the start of the first bus. Since we consider a feeder service, we also assume that there are no transfers between different bus lines of the DRFS, which means that each line is independent and can be optimized separately. Furthermore, bus lines with the same destination do not share bus stops, which means that passengers cannot choose which bus line to use. For this reason, only a single line is considered. In practice, multiple lines can be optimized in parallel, using multiple computational threads or different computers.

The performance of the DRFS has been optimized by using heuristics in [14]. The goal of this paper is to provide an exact optimization method to obtain optimal solutions for small and mid-size instances. These solutions can be used to assess the performance of the heuristic presented in [14]. We developed a Mixed Integer Problem (MIP) to optimize the performance of the service. However, it was found that the model could not be solved within a reasonable time for large, realistic instances. Therefore, to shorten the runtimes, two techniques are implemented: Separation of Sub-tour Elimination Constraints (SSEC) and Column Generation (CG). To assure the solutions found by the CG algorithm are optimal, a branch-and-price algorithm is needed.

In the next section, a literature review on public transport services is presented. In Section 3, the optimization model related to the DRFS and a mathematical model are presented. In Section 4, three different approaches to optimize the operation of the feeder service are presented. Section 5 discusses the results for several instances, obtained by optimizing the service. In the last section, conclusions are drawn and plans for future research are discussed.

2. Literature Review

The planning of Traditional Transport Services (TTS) happens in several stages, in which many strategical, tactical and operational decisions are taken ([15, 16]). Figure 1 illustrates this process. Clearly, the decisions in earlier stages influence the decisions in the next stages. Long-term, strategical decisions are taken in stage 1. This stage deals with the design of the infrastructure where, for example, the bus stop locations are defined and the type of vehicles is determined. Tactical, mid-term decisions are taken in stage 2 and stage 3. In stage 2, line planning and frequency setting are determined. Line planning consists of defining bus lines such that a given demand for transportation is satisfied. A bus line is a sequence of bus stops that is operated by a fleet of buses. Much research has been devoted to line planning, an extensive overview is given by Schöbel [17], and Iliopoulou et al. [18] review applications of meta-heuristics for line planning problems. Frequency setting determines how many buses travel along each line during a certain amount of time. In stage 3, the exact arrival times of the buses at every bus stop of the bus line are determined [10]. Decisions on an operational level are made in stage 4 and stage 5. In stage 4, vehicle scheduling defines the assignment of vehicles to bus lines in such a way that all planned trips can be fulfilled. In stage 5, duty scheduling defines the assignments of drivers to vehicles. Finally, in stage 6, real time control deals with any unpredictability during operation, like traffic jams or strikes for example. These stages are increasingly integrated and jointly optimized over the years, in order to improve the global planning as a whole. Carosi et al. [19] integrate and solve timetabling and vehicle scheduling with a meta-heuristic. Schöbel [20], on the other hand, develops an algorithmic scheme to optimize line planning, timetabling and vehicle scheduling together.

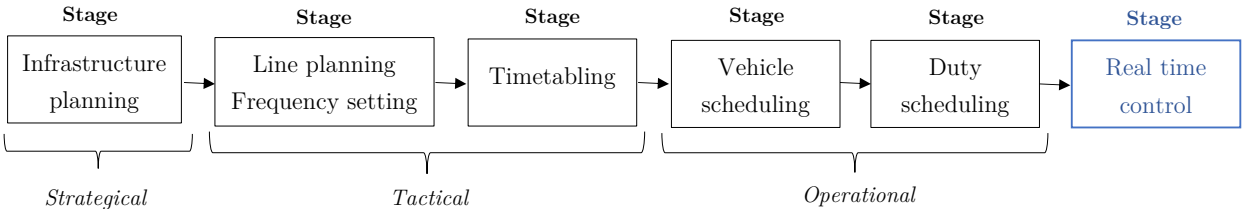


Figure 1: Planning in TTS

The popularity of On-Demand Transport Services (ODTS) has been rising over the years, with several services and models being developed in the last decades. Dynamic ride-sharing services aim to bring together travelers with similar itineraries and time schedules. Agatz et al. [21] give a review on different operations research models that have been developed in the literature. Classic on-demand services are ordered through a phone call. The Dial-A-Ride Problem (DARP) is a prime example of an optimization model for such services. In the DARP, vehicle routes need to be defined in order to pick-up and drop-off a number of passengers, with the objective to minimize the total cost of transportation. Generally speaking, pick-ups and drop-offs can take place anywhere in the route. This is called a door-to-door service. Furthermore, passengers often require to be picked up or dropped off during a specific time window. An extensive overview of the many variants of the DARP are given by Molenbruch et al. [22]. ODTS also consist of services that are not meant for public transport, for example the School Bus Routing Problem (SBRP). Here, a list of possible bus stops is given as well as a certain number of buses. Each student is assigned to one bus stop and one bus. The objective is to transport students to school while optimizing the bus routes that visit the assigned bus stops [23]. The SBRP can be seen as a vehicle routing problem with stop assignment, which adds an extra layer of complexity to the problem. In the literature, additional constraints are often considered as well, such as bus route scheduling or school bell adjustment. The former specifically considers time windows for arriving at schools and the latter considers the start time or end time of the school.

To the best of our knowledge, the service that is proposed in this paper has not been studied before. However, similar services that combine some characteristics of ODTS and TTS exist in the literature and have been gaining attention. For example, passengers can be picked up either from their homes or from serviced bus stops, and routes can be either partially or completely fixed. In some cases, one vehicle is available to serve a dedicated low demand area to bring the passengers to a terminal stop of the fixed-route public transport service [24]. In the USA, such services have been successfully implemented in recent years [25]. A relevant example of a service that integrates characteristics from TTS and ODTS is the Mobility Allowance Shuttle Transit (MAST) service. Quadrifoglio et al. [26] study a MAST service of the many-to-many type. In this service, vehicles have a fixed set of bus stops they always need to visit, i.e., a fixed path, and these stops also have fixed timetables. However, the vehicles may deviate from the fixed path. The customers that are served outside of the fixed path are served at their desired location and need to be within a certain radius from the fixed path in a so-called "zone". This service combines the high flexibility of door-to-door services with a fixed main route. Similarly, Zhao et al. [27] study a MAST service of the many-to-many type, where a bus always travels from one terminal to another and serves requests within a predetermined zone. This service is optimized with a non-backtracking nearest insertion algorithm, to obtain a dynamic routing and scheduling of the buses. In "customized bus" services, the planning of the service is an iterative and interactive process strongly involving the potential passengers. Stops, lines, timetables, etc., are proposed by an operator and then modified until passengers are satisfied [28]. In "variable-type" services, a classical service operates during peak hours and an on-demand service, where skipping fixed routes or portions of a fixed route is possible, operates during low-demand periods [29]. In the 'variable-type" or "customized bus"

services, the service provider decides beforehand where, when and how a bus will respond to customer requests. Both services ignore additional information and communication possibilities that are present in a smart city. Fu et al. [30] implement a real-time scheduling model with dynamic stop skipping. However, this model also limits itself to optimizing the schedule of the vehicles just before departure from the depot.

A bus service is called a feeder service when it is used to transport passengers from a, typically low demand, area to a transportation hub. Traditional feeder services have predetermined stops, routes, and timetables. In these traditional feeder services, the demand is considered to be known and is often derived from historical data. An important problem in the planning of feeder services is the Feeder Bus Network Design Problem (FBNDP). The FBNDP determines the design of a set of feeder bus routes, as well as the service frequency of each route for a period of time [31]. Ciaffi et al. [32] solve the FBNDP using a heuristic algorithm to generate routes, together with a genetic algorithm to find the best possible network of routes and their frequencies. This solution approach is implemented in two real-size networks as well, namely in Winnipeg and Rome. Lin et al. [33] present a multi-objective programming approach to solve the FBNDP, where route length and the maximum route travel time are minimized, while the service coverage is maximized simultaneously. This model is used for a case study of a metro station in Taichung City, Taiwan. Traditional feeder services, however, have their limitations, such as lack of accessibility and flexibility. It is inconvenient for some passengers, such as children, disabled, or senior passengers to reach one of the limited number of bus stops served by traditional feeder lines. Often it is also difficult to accommodate the different desired arrival times of the passengers[34].

Sun et al. [35] present a mixed-integer linear programming model for a demand-responsive feeder service similar to a DAR problem. The model is solved using a heuristic algorithm and is used in a case study in Nanjing City, China. The concept of MAST has been applied to feeder services as well. Lu et al. [36] develop a three-stage heuristic algorithm to optimize such a problem, together with a bus assignment sub-problem. Furthermore, Qiu et al. [37] analyze a feeder service similar to a MAST service, which is implemented in Salt Lake city in the USA. The authors conclude that the service has an advantage with respect to a fixed service under certain environmental circumstances. Another type of feeder service with flexible characteristics is the so-called Demand Responsive Connector (DRC). When and where to use the DRC rather than a fixed feeder service is further discussed by Li and Quadrifoglio [11]. In the DRC, no mandatory stops are considered and buses transport passengers from their origin location to transfer hubs within a predefined service area. In both the DRC and the MAST feeder service, passengers who need a ride are required to make a request for transportation before the vehicle departs, typically one to two hours in advance, and the planning cannot be changed during operation. Feeder services with flexible characteristics, like MAST and the DRC, have more success in low demand areas with a sparse population, while TTS services on the other hand thrive in high-demand and densely populated areas. Furthermore, research has suggested that an appropriate integration of these services could enhance mobility and increase the use and efficiency of public transport [38].

Similar to feeder services, Last-Mile Transportation (LMT) refers to services that transport passengers from a transportation hub, such as a city or a train station, to their homes or offices. The LMT problem is essentially the same transportation problem as a feeder service transportation problem, where the travel direction is reversed. Raghunathan et al. [39] determine passenger itineraries in order to minimize total transit time for all passengers, with each passenger arriving at the destination within a specified time window. The itineraries are determined by formulating the LMT as an integer linear programming problem and solving it with the use of a constructive heuristic. Numerical examples demonstrate that high quality solutions are obtained in acceptable

computational times. Markovic et al. [40] also study a LMT service. In this service, available vehicles are only dispatched when the number of passengers onboard reaches or exceeds a certain threshold. The objective is to maximize the system's profit, i.e., the revenue from fares minus the operational costs. A simulation model is employed to sample the passengers' arrivals at the terminal. An insertion algorithm provides an initial solution for the underlying traveling salesperson problem, thereafter a 2-opt procedure improves the solution. The system is tested on numerical examples, which include realistic cost estimates for Washington metropolitan area, with different demand levels.

The DRFS presented in [14], which is used for this paper, resembles a MAST service, such as the services described in Quadrifoglio et al. [26], Zhao et al. [27] and Qiu et al. [37], the most. The DRFS has a fixed route where it can deviate from, just as in MAST services. The main difference is that MAST services provide a door-to-door service to some customers within a certain radius, while the DRFS groups and assigns passengers at a number of bus stops. This increases the efficiency of the bus assignment and the routing. The timetable for the fixed route is also predefined in MAST services, limiting the time they can devote to deviating from the main route to provide the door-to-door service. This is not the case for the DRFS. Furthermore, in contrast to the DRFS, most MAST optimization models do not consider the capacity of the buses.

3. Problem description

In this section, the Demand-Responsive Feeder Service (DRFS) is described in detail. First, the feeder service is explained and the setting of the problem is determined. Next, an optimization model is defined in more formal terms. Finally, a mathematical model of the optimization problem is presented.

3.1. Description of the demand-responsive feeder service

The feeder service is situated in a residential area, or in general, an area with low demand for transportation. The bus lines of this service are designated shuttle buses that bring the inhabitants of this residential area to transportation hubs or to a nearby city center, i.e., the destination for all passengers is the same. A single bus line is considered because all bus lines in the DRFS are independent, i.e., no bus stops are shared by different lines. The bus line is operated by a fleet of vehicles, which means that multiple buses with different schedules travel along the same line. The set of bus stops S of the bus line consists of: a set of N mandatory stops $F = \{m_0, m_1, m_{N-1}\}$ and a set of optional stops $O = \{o_{0,0}, o_{0,1}, , o_{M-2,K_{M-2}}, o_{M-1,K_{M-1}}\}$. The mandatory and optional stops of a single bus line are illustrated in Figure 2.

The mandatory stops need to be visited by each bus from the line. Mandatory stops can, for example, be placed along some highway or main road. The optional stops are grouped into different clusters. Typically, the optional stops in a cluster will be relatively close to each other and scattered across a small town or neighborhood close to the main road on which the mandatory stops are located. There are $M \leq N-1$ clusters and each cluster c_k has K_k stops. The number of stops in each cluster can vary from cluster to cluster. This implies that there are $|S| = N + \sum_{k=0}^{M-1} K_k$ bus stops in a bus line with N mandatory stops and M clusters. An optional stop can be written as $o_{k,l}$, where k is the index of the cluster and l is the index of the stop within the cluster c_k.

The buses always start at the first mandatory stop m_0 and end at last mandatory stop m_{N-1}. The buses visit the mandatory stops in the sequence of their index, i.e., a bus cannot visit stop m_i before visiting stop m_{i-1}. A bus route can deviate from the route along the mandatory stops and

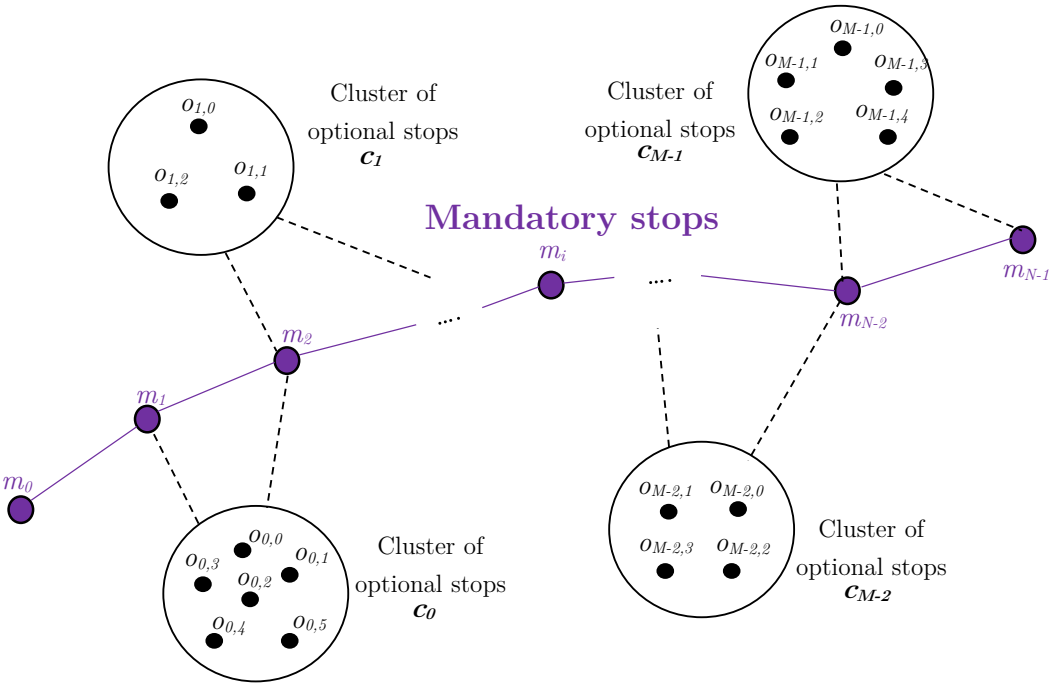


Figure 2: Bus stop structure of the demand-responsive feeder service

visit some optional stops in a cluster. From a cluster, the bus can travel to the next mandatory stop, to an optional stop in the same cluster, or to an optional stop in the next cluster. The main restriction is that all mandatory stops have to be visited in the correct order. The set of buses is labeled B.

It is assumed that a travel matrix is given, containing the travel times between each pair of bus stops, as well as a walking time matrix, containing the walking times between each passenger's origin location and all bus stops within walking distance. Furthermore, a dwell time coefficient, a deceleration time and a acceleration time are given as well. The deceleration and acceleration parameters are the amount of time a bus loses by slowing down to stop at a bus stop and speeding up to leave a bus stop respectively. The dwell time coefficient is the amount of time a bus loses by picking up a single passenger at a bus stop, i.e., if more passengers board the bus in the same bus stop, the dwell time at that bus stop increases. The total number of passengers requesting a ride during the planning horizon is $|P|$. Lastly, the capacity of the buses C, reserved for serving requests, is given as well. Although the service can still overcrowd if there are too many passengers without a reservation, the capacity constraint helps to control the crowding of the passengers with a reservation, which can alleviate crowding in general. Furthermore, crowding can be controlled by using "artificial capacities", which are smaller capacities than the actual capacity, in order to limit the number of reservations for a certain time period in the case we expect a large number of passengers without a reservation.

A passenger makes a request for a ride via a website, or a mobile application. The passengers state their origin location and desired arrival time at the destination m_{N-1}. This can happen until a given reservation deadline, for instance, one or two hours before the first bus of the fleet is dispatched from the depot, or the day before. After all requests are received, the route of each bus is optimized for the whole trajectory and the passengers are notified regarding the departure time of their bus and which bus stop they should go to for their journey. Since the requests are required to be known before the given deadline, the planning is optimized sufficiently in advance so passengers can walk to their departure stop well in time. After the planing is determined, also the

timetable of the buses at the mandatory stops is known and shared publicly. Potential passengers that did not make a request, but that are aware of this timetable information, can catch a bus at the mandatory stops or the optional stops being serviced. Potential passengers that are not aware of the request service or the timetable information, can still catch a bus at the mandatory stops at a relatively high frequency.

The assumption that the requests are known beforehand is not far-fetched since services have been implemented under these conditions before. An example of such a service is the "Belbus" from De Lijn (the Flemish regional transportation company) in Belgium [41], a DAR service that allows passengers to make a request for transportation until one day before operation. Another example is the flexible feeder service in Salt lake city in the USA [37] that requires a request for transportation before the service starts. Depending on the circumstances, a service that accepts real-time requests can be preferable, but the planning of such a service is left for future research. It needs to be noted, however, that the planning is considered "static" but not "fixed". This means that the routes and timetables of the buses are determined before the first bus starts, for a certain period of time. However, the routes and timetables will vary from period to period based on requests in each period.

To simplify the problem to some extent, the following assumptions are made. Passengers are assumed to travel alone. A group of passengers that want to travel together can make a set of identical requests, i.e., the same origin location and desired arrival time. However, it is not guaranteed that all passengers will be assigned to the same bus. Since there are no transfers between bus lines and no bus stops are shared between bus lines, as is typical for feeder services, the DRFS is optimized for a single bus line. We assume the DRFS is used to transport passengers from a rural or suburban area to a high demand area. In such cases, only a limited number of bus lines are present. For example, Wei et al. [42] optimize a fully flexible feeder service and apply this to a real life case in Chongqing China, where a single transportation hub is fed by three different neighborhoods.

One of the shortcomings of the DRFS is that the return trips from the transportation hub to the different stops are not explicitly considered. It is easy to see that planning the return trips results in the same optimization problem. Another shortcoming is that the DRFS does not guarantee a certain frequency of buses departing from the mandatory stops. This should, however, not be a problem if enough buses are utilized. These shortcomings will be addressed in future research, as the addition of these elements complicates the solution process considerably and is thus outside of the scope of this paper.

3.2. Optimization model

Different interdependent decisions need to be made in the DRFS optimization problem. All passengers need to be assigned to a departure bus stop, taking into account the walking time matrix. Consequently, each passenger also needs to be assigned to a bus that will bring him or her to the destination on time. The routing of each bus needs to be determined based on the optional bus stops that are assigned to passengers and that are selected for each bus. Furthermore, the departure time of each bus at the depot needs to be determined. This departure time will then determine the departure time at each bus stop. All these decisions are intertwined and affect one another. This problem can be viewed as an integration of a routing problem, an assignment problem, and a timetabling problem. Additionally, it needs to be noted that passengers are not always assigned to their closest bus stop. The passenger-bus-stop assignment determines which bus stops are chosen for the passengers in order to reduce the overall objective function value. For example, the travel times of the buses could increase significantly if all passengers are assigned to their closest bus stop. In that case, it might be more efficient to group passengers and assign some passengers to their second or third closest bus stop in order to reduce the travel time of the buses and consequently

the overall objective function value.

These decisions are also subject to a number of restrictions. First, the buses can have different routes according to the demand for transportation but all mandatory stops need to be visited by each bus. Secondly, the capacity of the buses is limited; no more passengers can be accepted in a bus if this capacity is reached. Thirdly, the passengers are not allowed to walk longer than a certain amount of time from their origin location to their assigned departure bus stop. Fourth, given that the requests of the passengers are known before the optimization process starts, it is assumed that all requests need to be served. As a last restriction, all passengers need to arrive within a certain time window. This means each passenger cannot arrive a certain amount of time earlier or later than his desired arrival time. Arriving too early or too late is not allowed and arriving earlier or later than the desired arrival time is penalized in the objective function. The arrival times of the passengers are thus part of both the constraints and the objective function. This last restriction ensures that the bus arrives at the destination before the latest and after the earliest desired arrival times of all passengers onboard. The restriction on arriving too early ensures that the passengers do not arrive extremely early at their destination, e.g. two hours early, on a differently scheduled bus. Arriving earlier implies departing earlier as well, which means that passengers may have to cancel or cut short a previous activity. It needs to be noted, however, that due to the arrival time constraints, the problem can become infeasible if the desired arrival times are spread out in time too much and not enough buses are available.

The objective of this problem is to optimize the service quality of the feeder service. This is modeled by minimizing a weighted sum of three factors. Firstly, the travel time for all buses is minimized. Shorter travel times for the buses will reduce the operating costs and the ride times for the passengers and thus improve the service quality. Secondly, the total walking time from the origin location of each passenger to the departure bus stop needs to be as low as possible. Thirdly, the absolute time difference between the desired arrival time and the actual arrival time at the destination, of each passenger, needs to be as close to zero as possible. For example, if a passenger requests to arrive at 18:00 at the destination and the bus is planned to arrive at 18:10, then the difference of 10 minutes is added to the objective function. The weights of this sum can be determined depending on the situation and on the preferences of the service provider. Although it is unlikely, giving objective function weight W_3 a high value can result in situations where passengers are assigned to a bus that travels longer and/or assigned to a bus stop farther away in order to make variable a_p^{early} smaller and reduce the objective function value. If late arrivals are considered more important than early arrivals, the service provider can simply introduce a fourth objective function weight W_4 for a_p^{late}. By giving W_4 a higher value than W_3, one can avoid unfavorable passenger assignments. In this paper it is assumed that both types of arrivals, i.e., late and early arrivals, are given equal priority and only three objective function weights are used.

The waiting time between the arrival of the passengers at their departure bus stop and the scheduled departure time of the bus at that bus stop is not considered. In this model, it is assumed that there are no significant delays and that the passengers will make a request beforehand. This means passengers can plan to arrive at the bus stop on time without incurring longer waiting times. Furthermore, the travel time of each bus is chosen over the travel time of each passenger. This is done in order to optimize the routes of the buses rather than the onboard time of each passenger individually. Optimal bus routes, however, also imply shorter travel times for passengers onboard. As discussed in Section 1, the onboard times are typically less important to customers compared to the reliability of the service and arrival time at their destination. Service reliability is inherently part of this service as it deals with passenger requests and a timely arrival at the destination is covered by the third component of the objective function.

3.3. Mathematical Model

For the sake of clarity, the notations of the different sets, parameters and decision variables are listed in Table 1.

The objective z of the mixed integer programming model is to minimize a weighted sum of three terms related to the service quality. The parameters W_1, W_2 and W_3 are weights given to these terms and can be determined by the user. The first term (1) calculates the travel time of each bus. It includes the acceleration and deceleration times of the buses when they leave a bus stop and when they stop at a bus stop respectively. The dwell time of the buses at a bus stop is also taken into account and is dependent on the number of passengers that are picked up at the stop. The second term (2) minimizes the walking time of each passenger from their origin location to their assigned departure bus stop. The last term (3) is the time each passenger will arrive at the destination before or after his or her desired arrival time.

Min

$$z = W_1 \left[\sum_{b \in B} \sum_{i \in S} \left(\sum_{j \in S} (T_{ij}^{\mathrm{t}} + T^{\mathrm{a}}) x_{bij} + T^{\mathrm{d}} \sum_{p \in P} y_{pbi} \right) \right] \tag{1}$$

$$+ W_2 \left[\sum_{b \in B} \sum_{i \in S} \sum_{p \in P} T_{pi}^{\mathrm{w}} y_{pbi} \right] \tag{2}$$

$$+ W_3 \left[\sum_{p \in P} \left(a_p^{\mathrm{late}} + a_p^{\mathrm{early}} \right) \right] \tag{3}$$

The first group of constraints deals with the routing of the buses. Constraints (4) ensure that, for optional bus stops, at most one arc enters or leaves any stop. Constraints (5) ensure that, for each mandatory stop, exactly one arc enters or leaves. If an arc enters the stop, there must be an arc leaving the stop and vice versa (6). The only exceptions are m_0, where exactly one arc leaves and none enter, and m_{N-1}, where exactly one arc enters and none leave. Constraints (7) and constraints (8) ensure that no bus ever has stop m_0 as a successor or stop m_{N-1} as a predecessor. Constraints (9) are cycle elimination constraints.

$$\sum_{j \in S} x_{bij} \leq 1 \qquad \forall\ i \in O,\ b \in B \tag{4}$$

$$\sum_{j \in S} x_{bij} = 1 \qquad \forall\ i \in F,\ b \in B \tag{5}$$

$$\sum_{l \in S} x_{bil} = \sum_{l \in S} x_{bli} \qquad \forall\ i \in S \setminus \{0, N-1\},\ b \in B \tag{6}$$

$$\sum_{i \in S} x_{bi0} = 0 \qquad \forall\ b \in B \tag{7}$$

$$\sum_{i \in S} x_{bN-1i} = 0 \qquad \forall\ b \in B \tag{8}$$

$$\sum_{i \in S_t} \sum_{j \in S_t} x_{bij} \leq |S_t| - 1 \qquad \forall\ S_t \subset S, S_t \neq \emptyset, b \in B \tag{9}$$

A second group of constraints deals with capacities or threshold values. Constraints (10) ensure that no passenger needs to walk more than a predefined maximum value D^{w}. Constraints (11)

Sets	
B	Set of buses
S	Set of all bus stops
O	Set of optional bus stops
F	Set of mandatory bus stops
P	Set of passengers using the service during the optimization horizon
Parameters	
K_k	Number of optional bus stops in cluster k
M	Number of clusters
T_{ij}^{t}	Travel time from bus stop $i \in S$ to bus stop $j \in S$
T^{d}	Dwell time per passenger boarding
T_{pi}^{w}	Walking time of passenger $p \in P$ to departure bus stop $i \in S$
T_p^{arr}	Desired arrival time of passenger $p \in P$ at the destination bus stop m_{N-1}
T^{a}	Average acceleration and deceleration time of a bus
D^{w}	Maximum value for individual walking time
D^{late}	Maximum value for arriving late
D^{early}	Maximum value for arriving early
C	Capacity of the buses
W_1	Relative weight given to the travel time of the buses
W_2	Relative weight given to the walking time of the passengers
W_3	Relative weight given to the absolute difference in desired and actual arrival time of the passengers
M_0, M_1	Large natural numbers for the Big M constraints
Decision Variables	
x_{bij}	0-1 variables determining if bus $b \in B$ visits bus stop $j \in S$ immediately after visiting bus stop $i \in S$
y_{pbi}	0-1 assignment variables which assume value 1 if passenger $p \in P$ is assigned to bus $b \in B$, and departure bus stop $i \in S$
a_p	Arrival time of passenger $p \in P$ at destination bus stop m_{N-1}
a_b^{bus}	Arrival time of bus $b \in B$ at destination bus stop m_{N-1}
d_b	Departure time of bus $b \in B$ at the first mandatory bus stop m_0
a_p^{late}	$a_p - T_p^{\mathrm{arr}}$ when passenger $p \in P$ is late
a_p^{early}	$T_p^{\mathrm{arr}} - a_p$ when passenger $p \in P$ is early

Table 1: Notation for the MIP

regulate the number of passengers on each bus, so that buses cannot transport more passengers than a given capacity. Constraints (12) and (13) ensure that all passengers arrive within the time window $\left[T_p^{\text{arr}} - D^{\text{early}}, T_p^{\text{arr}} + D^{\text{late}}\right]$.

$$y_{pbi} = 0 \qquad \forall\ p, i | T_{pi}^{\text{w}} > D^{\text{w}},\ b \in B \quad (10)$$

$$\sum_{p \in P} \sum_{i \in S} y_{pbi} \leq C \qquad \forall\ b \in B \quad (11)$$

$$a_p^{\text{late}} \leq D^{\text{late}} \qquad \forall\ p \in P \quad (12)$$

$$a_p^{\text{early}} \leq D^{\text{early}} \qquad \forall\ p \in P \quad (13)$$

Constraint (14) defines the decision variables a_p^{early} and a_p^{late} as positive deviations between the actual arrival a_p time and the desired arrival time T_p^{arr} of the passengers. Given the objective function, one of the two variables a_p^{late} or a_p^{early} will be zero for each passenger p in the optimal solution.

$$T_p^{\text{arr}} - a_p + a_p^{\text{late}} - a_p^{\text{early}} = 0 \qquad \forall\ p \in P \quad (14)$$

Constraints (15) and (16) define the variables a_b^{bus} and a_p, respectively the arrival time of a bus b and a passenger p at the destination. Constraints (16) are big M constraints that link the passengers to the buses. The left-hand side of constraints (16) define a_p, in case the passenger arrives late or just on time, i.e., if $a_p - T_p^{\text{arr}} \geq 0$. The right-hand side of constraints (16) define a_p, in the case the passenger arrives early, i.e., if $a_p - T_p^{\text{arr}} < 0$. Constraints (17) are big M constraints that link the y and x variables. A passenger with a certain departure bus stop i is only assigned to a bus that visits bus stop i. This constraint is only valid for the optional stops since the mandatory stops are always visited. If these stops were included in this constraint, it would always assign some people to the mandatory stops.

$$a_b^{\text{bus}} = \sum_{i \in S} \left(\sum_{j \in S} (T_{ij}^{\text{t}} + T^{\text{a}}) x_{bij} + T^{\text{d}} \sum_{c \in P} y_{cbi} \right) + d_b \qquad \forall\ b \in B \quad (15)$$

$$a_b^{\text{bus}} - \left(1 - \sum_{i \in S} y_{pbi}\right) M_0 \leq a_p \leq a_b^{\text{bus}} + \left(1 - \sum_{i \in S} y_{pbi}\right) M_0 \qquad \forall\ b \in B,\ p \in P \quad (16)$$

$$\sum_{p \in P} y_{pbi} \leq M_1 \sum_{l \in S} x_{bil} \qquad \forall\ i \in O,\ b \in B \quad (17)$$

Lastly, a set of constraints ensures that every passenger is assigned to exactly one bus and one departure bus stop.

$$\sum_{b \in B} \sum_{i \in S} y_{pbi} = 1 \qquad \forall\ p \in P \quad (18)$$

The remaining constraints determine the domains of the variables.

$$y_{pbi} \in \{0, 1\} \qquad \forall\ b \in B,\ b \in B,\ p \in P \quad (19)$$

$$x_{bij} \in \{0, 1\} \qquad \forall\ b \in B,\ i \in S,\ j \in S \quad (20)$$

$$a_p,\ a_p^{\text{late}},\ a_p^{\text{early}} \in \Re^+ \qquad \forall\ p \in P \quad (21)$$

$$a_b^{\text{bus}},\ d_b \in \Re^+ \qquad \forall\ b \in B \quad (22)$$

4. Solution methods

In this section, solution methods to solve the optimization model defined in Section 3.2 are described. First, the model is solved using a commercial solver. Then, two improvements are developed in order to reduce the runtime. The first improvement is a separation of constraints, that is utilized to eliminate cycles. The second improvement is a combination of column generation and separation of constraints.

4.1. Using CPLEX to solve the MIP

The problem, mathematically modeled in Section 3.3, is solved using the commercial solver CPLEX. Even though this is essentially a black-box procedure, it needs to be noted that the choice of M_0 and M_1 i.e., the large constant values in the model, have a significant impact on the time required to solve the MIP. If these "big M's" are too large, the MIP will have a weak linear relaxation and consequently, a weak lower bound. A weak lower bound in turn, makes the branch-and-bound search that is required to solve this MIP with CPLEX much more time consuming. For this reason, the big M's in these constraints need to be chosen with the lowest possible value while still being larger than any possible value on the left-hand side of the inequality of these constraints [43]. To this end we choose the values for the big natural numbers M_0 and M_1 as:

$$M_0 = \max_{p \in P} T_p^{\text{arr}} + D^{\text{late}} - \left(\min_{p \in P} T_p^{\text{arr}} - D^{\text{early}} \right) \tag{23}$$

$$M_1 = C \tag{24}$$

The reasoning behind formula (23), which sets the value for M_0, is the following. In the case a passenger p is not assigned to bus b, the left-hand side of constraints (16) will be equal to the difference of the arrival time of passenger p and the arrival time of bus b, otherwise they will equal to zero. When a passenger is not assigned to a bus, the largest possible difference is the range of the time window any bus can arrive at the destination. This is the difference between a bus arriving D^{early} too early for the passenger that needs to arrive the earliest and a bus arriving D^{late} too late for the passenger that has the latest desired arrival time. The large number M_1 has a value equal to the capacity of the buses (24), since the left-hand side of constraint (16) counts the number of passengers that board bus b at bus stop i.

4.2. Separation of sub-tour elimination constraints

In integer programming formulations that involve routing, Sub-tour Elimination Constraints (SECs) are used to ensure connectedness of the routes. The number of SECs is exponential in the size of the considered problem instance, since there are 2^n SECs in a graph with n nodes. As a result, the runtime of such a model increases considerably. A common strategy to tackle this issue is to solve those SEC formulations with the so-called "lazy constraints" at integer nodes in the branch-and-bound tree. Here, violated SECs are identified dynamically in the course of the branch-and-bound algorithm and are subsequently added to the formulation. This identification process is commonly referred to as separation [43]. The Separation of violated SECs (SSEC) is usually performed with as few lazy constraints as possible [44]. This strategy is implemented in this research as well. In this model, the "sub-tours" are technically cycles because a valid solution is a point-to-point route, rather than a tour. In the SSEC, the model will be solved as usual using CPLEX but without the cycle elimination constraints (9). At every instance where CPLEX finds a solution that has integer values, i.e., a feasible solution ignoring the cycle elimination constraints, the cycle identification algorithm is executed. In this algorithm, cycles are identified and the corresponding constraints are added in order to prevent these cycles. To dynamically identify and add the cycle elimination constraints to the model, a straightforward algorithm is used.

It needs to be noted that the addition of fractional cuts could improve the algorithm by decreasing the size of the feasible solution space. This would, in turn, decrease the runtime by improving the lower bounds which makes it easier to prove optimality. However, this is a different solution approach that requires a completely different implementation. Since the MIP is mainly slowed down by the exponential number of SECs, we opted to implement the SSEC rather than the insertion of fractional cuts. Furthermore, the potential lower bound issue is solved with the use of column generation, as it is explained in Section 4.3.

4.3. Column Generation

As will be shown in Section 5.2, the SSEC greatly improves the runtime for some instances. However, although the approach manages to find optimal or very good solutions quickly, it still produces solutions with large integrality gaps due to poor lower bounds. In order to remedy this issue, Column Generation (CG) can be utilized. In CG, only a subset of variables is considered when solving the problem and this can greatly reduce the computational time. The original MIP needs to be reformulated as a Master Problem (MP), which is a column-wise formulation of the original MIP. The MP is constructed in such a way that selecting the right columns, i.e., selecting which variables are non-zero in the optimal solution, optimizes the original MIP. The MP is then split into two problems; called respectively the Restricted Master Problem (RMP) and the Sub-Problem (SP). The RMP is the same as the MP, with the difference that only a subset of columns are included. The SP, on the other hand, is a new problem created to identify a new column corresponding to a new variable.

In what follows, the master problem, the algorithm for the construction of the initial solution, the RMP and the SP are discussed in more detail.

4.3.1. Master problem and optimality

The original MIP of the DRFS can be reformulated as a MP as follows. The service provider has an unlimited number of buses that each have a predetermined timetable, route and a number of passengers onboard, and the problem is determining which buses are chosen. In this formulation, the columns correspond with the buses, i.e., adding a new column to the MP is the same as adding an additional bus. Each bus is subject to the same constraints as the buses in the original MIP and the same objective is optimized.

The MP is given below. The variables λ^b are binary variables that determine whether a bus b is selected or not. The MP has two constraints: exactly $|B|$ buses are utilized (25), and each passenger must be transported to the destination and is assigned to exactly one bus (26). The objective function of the MP is the same as the original MIP, with the difference that there are now several buses with different sets of passengers onboard. The set of all possible buses with passengers onboard is denoted as B_T

Minimize: $z_{MP} =$

$$\sum_{b\in B_T}\left(\sum_{i\in S}\sum_{j\in S} W_1\left(T^{\mathrm{t}}_{ij}+T^{\mathrm{a}}\right)x^b_{ij}+\sum_{i\in S}\sum_{p\in P}\left(W_1T^{\mathrm{d}}+W_2T^{\mathrm{w}}_{pi}\right)y^b_{pi}+\sum_{p\in P}W_3\left(a^{\mathrm{late},b}_p+a^{\mathrm{early},b}_p\right)\right)\lambda^b$$

S.t.

$$\sum_{b\in B_T}\lambda^b = |B| \tag{25}$$

$$\sum_{b\in B_T}\sum_{i\in S} y^b_{pi}\lambda^b = 1 \qquad \forall\ p\in P \tag{26}$$

$$\lambda^b \in \{0,1\} \qquad \forall\ b\in B_T \tag{27}$$

It needs to be noted that column generation generally yields optimal solutions for the integrally relaxed master problem and not for the original master problem. Under normal circumstances, this would mean that this approach is an approximate method because there is no guarantee that the solution of the relaxed MP is integer. To guarantee optimal solutions, the CG algorithm needs to be embedded into a Branch-And-Price (B&P) algorithm. The branch-and-price algorithm was first presented in 1981 by Ryan and Foster [45] as the well-known Ryan-Foster branching rule. The authors used this algorithm to solve a bus routing problem. B&P combines CG with a branch-and-bound algorithm. In the first step of the B&P algorithm, the relaxed RMP is solved with column generation. If the solution to the relaxed RMP is integral, the optimal solution is found and the algorithm stops. Otherwise, the algorithm continues by branching on a variable of the RMP and by generating more columns. This process is continued until an integer optimal solution is found. It needs to be noted, however, that the RMP that is presented in this paper produces integral solutions for all the instances that are tested. This means that the optimal solution is found before any branching is needed and the CG algorithm alone yields optimal solutions. For this reason we limit ourselves to describing the column generation algorithm without the branching procedure. In the case there is an instance where a non-integer solution is found, the branching rules provided by Vanderbeck and Wolsey [46] can be used to find the optimal solution. Vanderbeck and Wolsey [46] provide well performing branching rules for binary constraint matrices, such as the constraint matrix of our RMP.

4.3.2. Construction of an initial solution

The initial solution is constructed by iteratively constructing a solution for each bus b separately. All passengers are put in a list L_p and are sorted according to their desired arrival time T_p^{arr}. The passengers are then added to each bus $b \in B$ in this order. Passengers that are assigned to bus b are added to list L_b and removed from list L_p. The algorithm will stop adding passengers to a bus b either when the bus reaches its capacity or when $\max_{p \in L_b} T_p^{\text{arr}} + \min_{p \in L_b} T_p^{\text{arr}} < D^{\text{late}} + D^{\text{early}}$. The last expression ensures that all passengers reach the destination within their time window. In each bus b, all passengers are assigned to the closest bus stop to their location, as their departure bus stop. Afterwards, the route of bus b is determined. To build the bus route, all the mandatory stops are added to the route first. Next, all the optional stops that are assigned to passengers assigned to bus b are inserted in the existing route, between the two closest mandatory stops but further in an arbitrary manner. In the last step of the construction of the initial solution of bus b, the departure time of each bus is determined. Since the route of the bus is already determined, its departure time at each stop and the arrival time at the destination follow directly from the departure time from the first mandatory bus stop. The bus needs to arrive at the destination within the given time window $\left[\max_{p \in L_b} T_p^{\text{arr}} - D^{\text{early}}, \min_{p \in L_b} T_p^{\text{arr}} + D^{\text{late}}\right] = [LB_b, UB_b]$. Here, UB_b and LB_b denote the upper and lower bounds of the interval, for bus b. The arrival time of the buses is determined in such a way that the arrival time of the passengers is optimized. This is expressed in the objective function as $\sum_{p \in L_b} \left| T_p^{\text{arr}} - a_p \right|, \forall b \in B$. This is a sum of absolute deviations, and as shown by Dodge [47], the median, in this case the median of the desired arrival times of the passengers onboard the bus, minimizes this sum. However, an additional constraint is that the solution must be within the interval $[LB_b, UB_b]$. Whenever the median is larger than UB_b the arrival time will be set to UB_b. When the median is smaller than LB_b, the arrival time will be set to LB_b. The correction of the arrival time still gives the best possible solution since $\sum_{p \in L_b} |T_p^{\text{arr}} - a_p|$ is a convex function of a_p. If the departure time at the first mandatory stop that follows from the optimal arrival time at the last mandatory stop is negative, the arrival time at the destination is set to UB_b. If this still leads to a negative departure time at any stop, the instance is infeasible and the algorithm stops. For the next bus, the remaining passengers in the list are considered and the construction of the

solution for the next bus starts. This construction process is repeated until there are no passengers left in the list or all available buses have been used.

4.3.3. Restricted master problem

The RMP for CG is given below. The variables λ^b are binary variables that determine whether a bus b is selected or not. In the RMP these variables are integrally relaxed. The objective function and the constraints of the RMP are the same as the MP, with the difference that only a subset of these buses are selected by the RMP. In the RMP, new columns, i.e., new buses generated by the SB, are added iteratively. The number of buses that are present in the RMP at iteration it is B_{it}. It needs to be noted that in the MP and in the RMP, the only variables are λ^b. The variables in the original model are parameters here. The dual prices corresponding with constraints (28) and (26) are σ and $\pi_p,\ \forall p \in P$ respectively.

Minimize: $= z_{RMP}$

$$\sum_{b<B_{it}} \left(\sum_{i\in S}\sum_{j\in S} W_1 \left(T^{\mathrm{t}}_{ij} + \delta\right) x^b_{ij} + \sum_{i\in S}\sum_{p\in P} y^b_{pi} \left(W_1 T^{\mathrm{d}} + W_2 T^{\mathrm{w}}_{pi}\right) + \sum_{p\in P} W_3 \left(a^{\mathrm{early},b}_p + a^{\mathrm{late},b}_p\right) \right) \lambda^b$$

S.t.

$$\sum_{b<B_{it}} \lambda^b = |B| \tag{28}$$

$$\sum_{b<B_{it}} \sum_{i\in S} y^b_{pi} \lambda^b = 1 \qquad \forall\ p \in P \tag{29}$$

$$0 \le \lambda^b \le 1 \qquad \forall\ b < B_{it} \tag{30}$$

4.3.4. Subproblem

The subproblem is given below. The SP calculates the route, timetabling and passenger assignment of a single bus, i.e., a column of the RMP. In the objective, the reduced cost of the new bus is calculated using the dual prices σ and π_p. The SP has the same set of variables as in the original problem, except for the fact that variables only pertain to a single bus and thus do not have the b index anymore. The objective function uses the dual prices of the RMP to calculate the reduced cost of this column. The constraints of the SP are the same as in the original problem. The SP excludes the cycle elimination constraints because it makes use of the same separation technique as in Section 4.2. Furthermore, some additional constraints (45) and (46), are added. These constraints are not necessary to ensure feasibility but they increase the computational efficiency by limiting the number of possible solutions. These big-M constraints ensure that when a passenger p is not picked up by a bus b, a^b_p will be equal to T^{arr}_p and this will make $a^{\mathrm{late},b}_p$ and $a^{\mathrm{early},b}_p$ equal to zero. Consequently, the passenger does not have an impact on the third component of the objective function. Due to the limits set in constraints (39) and (40), the big M numbers M_2 and M_3 are chosen as D^{late} and D^{early} respectively.

Minimize: $z_{SP} =$

$$\sum_{i\in S} \left(\sum_{j\in S} W_1 (T^{\mathrm{t}}_{ij} + \delta) x_{ij} + \sum_{p\in P} y_{pi} \left(W_1 T^{\mathrm{d}} + W_2 T^{\mathrm{w}}_{pi} - \pi_p\right) \right) + \sum_{p\in P} W_3 \left(a^{\mathrm{late}}_p + a^{\mathrm{early}}_p\right) - \sigma$$

S.t.

$$\sum_{\substack{j\in S\\ i<j}} x_{ij} \leq 1 \qquad \forall\ i \in O \quad (31)$$

$$\sum_{j\in S} x_{ij} = 1 \qquad \forall\ i \in F \quad (32)$$

$$\sum_{l\in S} x_{il} = \sum_{l\in J} x_{li} \qquad \forall\ i \in S_{0,N-1} \quad (33)$$

$$\sum_{i\in S} x_{i0} = 0 \quad (34)$$

$$\sum_{i\in S} x_{N-1i} = 0 \quad (35)$$

$$x_{ii} = 0 \qquad \forall\ i \in S \quad (36)$$

$$\sum_{i\in S} T^{\mathrm{w}}_{pi} y_{pi} \leq d \qquad \forall\ p \in P \quad (37)$$

$$\sum_{p\in P}\sum_{i\in S} y_{pi} \leq C \quad (38)$$

$$a_p^{\mathrm{late}} \leq D^{\mathrm{late}} \qquad \forall\ p \in P \quad (39)$$

$$a_p^{\mathrm{early}} \leq D^{\mathrm{early}} \qquad \forall\ p \in P \quad (40)$$

$$T_p^{\mathrm{arr}} - a_p + a_p^{\mathrm{late}} - a_p^{\mathrm{early}} = 0 \qquad \forall\ p \in P \quad (41)$$

$$a^{\mathrm{bus}} = \sum_{i\in S}\left(\sum_{j\in S}(T^{\mathrm{t}}_{ij} + T^{\mathrm{a}})x_{ij} + T^{\mathrm{d}}\sum_{c\in P} y_{ci}\right) + d \quad (42)$$

$$a^{\mathrm{bus}} - \left(1 - \sum_{i\in S} y_{pi}\right) M_0 \leq a_p \leq a^{\mathrm{bus}} + \left(1 - \sum_{i\in S} y_{pi}\right) M_0 \qquad \forall\ p \in P \quad (43)$$

$$\sum_{p\in P} y_{pi} \leq M_1 \sum_{l\in S} x_{il} \qquad \forall\ i \in O \quad (44)$$

$$a_p^{\mathrm{late}} \leq M_2 \sum_{i\in S} y_{pi} \qquad \forall\ p \in P \quad (45)$$

$$a_p^{\mathrm{early}} \leq M_3 \sum_{i\in S} y_{pi} \qquad \forall\ p \in P \quad (46)$$

$$y_{pi} \in \{0,1\} \qquad \forall\ i \in S, p \in P \quad (47)$$

$$x_{ij} \in \{0,1\} \qquad \forall\ i \in S, j \in S \quad (48)$$

$$a_p,\ a_p^{\mathrm{late}},\ a_p^{\mathrm{early}} \in \Re^+ \qquad \forall\ p \in P \quad (49)$$

$$a^{\mathrm{bus}},\ d \in \Re^+ \quad (50)$$

4.3.5. Stabilized column generation

It is apparent, in Section 5.2, that the CG approach needs a large number of iterations on certain instances in order to find the optimal solution. This is an indication that the solution process of CG is converging slowly. In order to improve the convergence of the CG algorithm, stabilization methods can be applied. These methods speed up the convergence of CG and consequently decrease the runtime. We opted to implement the stabilization method presented by du Merle et al. [48]. The stabilization procedure of the Stabilized Column Generation (SCG) algorithm starts by modifying the MP to create the Stabilized Master Problem (SMP). Surplus and slack variables γ_- and γ_+ are added to the constraints of the SMP. These variables are strictly positive and have upper bounds ϵ_- and ϵ_+. Furthermore, these variables are penalized in the objective function of the SMP by vectors δ_- and δ_+. This results in penalizing the dual variables when they lie outside of the interval $[\delta_-, \delta_+]$.

An additional stopping criterion for the SCG algorithm is that the variables γ_- and γ_+ need to be zero in order to guarantee the optimality of the original MP as well. During each iteration of the SCG ϵ_-, ϵ_+, δ_- and δ_+ need to be updated in order to ensure finite convergence of the algorithm. The updating strategy is to decrease ϵ_- and ϵ_+ so that they vanish in a finite number of iterations. This is done by starting off with a relatively large value and decreasing it with a small amount in each iteration of the SCG algorithm. Parameters δ_- and δ_+ are updated to $\tilde{\pi} - \xi$ and $\tilde{\pi} + \xi$ respectively, with $\tilde{\pi}$ the corresponding dual variable and $\xi > 0$. This update only takes place when a column with non-negative reduced cost is found by the SCG algorithm.

5. Results

In this section, the results that are obtained with the solution methods defined in Section 4 are discussed. First, a set of instances is generated to test the optimization model. Second, the runtimes of the different solution methods are compared, including the original MIP which is solved with CPLEX. Third, different experiments are conducted in order to determine which parameters of the instances contribute the most to the runtime of each solution method. Finally, a qualitative analysis is presented.

To reduce the complexity of the problem, unnecessary data is excluded in each instance. In particular, bus stops that are farther away than the maximum allowed walking time D^{w} from a passenger are excluded. Similarly, optional bus stops that are farther away than a mandatory bus stop can be ignored as potential departure bus stops for the passengers.

5.1. Test instances

To test the different models and algorithms, a set of instances is created. These instances are randomly generated and are listed in Table 2. The instances vary in number of buses $|B|$, the number of mandatory stops $|F|$, requests $|P|$ and bus stops $|S|$. Two further assumptions are that the number of optional stops per cluster K is the same in each cluster and that there is exactly one cluster in between two mandatory stops. In Table 2, the instances are ordered according to the number of binary variables V in the model. This parameter V is an indication of the size of the instances. The largest instance, I_{14}, is considerably larger than the rest of the instances. In our opinion, this last instance is large enough to be considered realistic in a rural or sub-urban setting.

For all instances, the parameters that are used are listed in Table 3. All instances, as well as the solutions discussed in this paper are available in detail online: `https://www.mech.kuleuven.be/en/cib/drbp#section-0`. The objective weights W_1, W_2 and W_3 are set to 0.25, 0.35 and 0.40 respectively. The values are chosen with the goal of giving each component of the objective function approximately equal importance. A detailed study of the effect of the weight values on the solutions is beyond the scope of this paper. Nonetheless, a small analysis on the impact of these weights is presented in this section.

Parameter	Value	Unit
C	15	passengers
D^{late}	300	s
D^{early}	900	s
D^{w}	1200	s
T^{a}	30	s
T^{d}	5	$\frac{\mathrm{s}}{\mathrm{passenger}}$

Table 3: Parameters

Instance	|B|	|F|	K	|S|	|P|	V
I_1	2	3	3	9	12	1188
I_2	3	3	3	9	12	1296
I_3	2	4	3	13	16	3120
I_4	3	4	3	13	16	3328
I_5	2	5	3	17	20	6460
I_6	3	5	3	17	20	6800
I_7	4	5	3	17	20	7140
I_8	3	6	3	21	25	12600
I_9	4	6	3	21	25	13125
I_{10}	3	6	3	21	30	15120
I_{11}	4	6	3	21	30	15750
I_{12}	4	7	3	25	30	21750
I_{13}	5	7	3	25	30	22500
I_{14}	5	10	3	37	40	62160

Table 2: Test instances of the DRFS

5.2. Comparison of the different approaches

The models are solved using CPLEX version 12.9 on a computer with a Windows 10 Enterprise operating system, an Intel Core^{TM} i7-8850H, 2.60Ghz CPU and 16 GB of RAM. All instances are solved with the original MIP, with the original MIP using the SSEC, and with the restated model using CG and SSEC and with the Stablized Column Generation (SCG) and SSEC. All runtimes are limited to one hour, i.e., all algorithms stop either after the optimal solution is found, or after one hour of runtime. The number of iterations that are needed for the convergence of both the CG and the SCG algorithms are shown as well. In case the optimal solution is not found after one hour, the best feasible solution up until that point in time is reported. If no feasible solution could be found after one hour, the objective function value is and the gap are denoted as "/" in Table 4. For the first two models, a gap is displayed as well. This is the ratio between the best feasible solution and the best lower bound found thus far. The gap of the last model is calculated by using the optimal solution as the lower bound. During the column generation process the best solution found thus-far is integral, thus there is no other lower bound available. The optimal solution is found by running the algorithm without any runtime limitation. In Table 4, the results of this experiment are displayed. Objective values that are not optimal after one hour of runtime, are displayed in italic.

It can be seen that the original MIP can only solve the problem to optimality until instance I_4. Larger instances cannot be solved with this model, within one hour. Furthermore, instances larger than instance I_7 are too large for the MIP to even find a gap value. This means that the model could not find a feasible solution for the instances in one hour of runtime. When separation is applied, the problem can be solved to optimality until instance I_6. The runtimes of the models also decrease considerably. The SSEC model did find the optimal solution for instance I_7, but was unable to prove that this solution is optimal and ended with a gap of approximately 14% after one hour. The column generation models could solve all instances except the last one to optimality within one hour. The models did manage to find a better integer feasible solution than the SSEC model for this instance. By looking at the number of needed iterations it can be seen that the SCG model performs quite well in comparison to the CG model. In general, 62% fewer iterations are needed for convergence when the CG is stabilized. This also results in significantly lower runtimes.

It needs to be noted that the best solution for the last instance found by the CG model after one hour is integer as well and no rounding was needed. Furthermore, the CG model did manage to find the optimal solution for the RMP in the last instance after a prolonged time and this solution was integer, which means that it is the optimal solution for the MP.

	CG + SSEC	*SCG + SSEC*	*MIP*	*SSEC*	*CG + SSEC*	*SCG + SSEC*	*MIP*	*SSEC*	*(S)CG + SSEC*	*MIP*	*SSEC*	*(S)CG + SSEC*
	Iterations		**Runtime (s)**				**Objective function value (s)**			**gap**		
I_1	51	27	1.125	0.313	4.247	2.541	3143	3143	3143	0.00%	0.00%	0.00%
I_2	51	36	3.987	19.21	3.878	3.014	2932	2932	2932	0.00%	0.00%	0.00%
I_3	131	35	26.41	1.047	26.63	7.874	4883	4883	4883	0.00%	0.00%	0.00%
I_4	98	54	1354	178.1	16.84	11.41	4447	4447	4447	0.00%	0.00%	0.00%
I_5	191	71	3600	2.425	75.87	27.64	*7434*	7294	7294	43.2%	0.00%	0.00%
I_6	154	63	3600	901.8	84.12	34.54	*6688*	6117	6117	37.4%	0.00%	0.00%
I_7	110	71	3600	3600	56.14	37.32	*6403*	5902	5902	32.0%	14.1%	0.00%
I_8	230	91	3600	3600	488.9	192.7	/	*7857*	7826	/	12.8%	0.00%
I_9	217	63	3600	3600	312.1	126.2	/	*7308*	7288	/	16.8%	0.00%
I_{10}	496	48	3600	3600	1819	131.5	/	*9489*	9484	/	4.50%	0.00%
I_{11}	488	77	3600	3600	1724	278.7	/	*8824*	8790	/	20.0%	0.00%
I_{12}	427	71	3600	3600	2316	388.4	/	*9071*	9021	/	21.3%	0.00%
I_{13}	284	85	3600	3600	1443	440.5	/	*8998*	8937	/	22.0%	0.00%
I_{14}	/	/	3600	3600	3600	3600	/	*12644*	*12543*	/	22.5%	1.50%

Table 4: Results of the instances for the different approaches

In Figure 3, runtimes for the different models are displayed on a logarithmic scale. The runtime of the original MIP increases exponentially and monotonically with the size of the instances. The runtime of the SSEC model also increases exponentially, but the increase is less steep and this allows the model to solve more instances. For the CG model, the increase in runtime is much less steep. This allows the model to solve almost every instance within one hour. The runtimes of the SSEC model does not increase monotonically. A possible cause for this non-monotonic increase can be found in the number of buses: the number of ways in which passengers can be assigned to buses increases steeply with the number of buses and this results in a less tight lower bound for the relaxed model. This, in turn, leads to larger gaps and to a longer search in the branch-and-bound tree to prove optimality. Furthermore, the column generation models only become faster than the other models when the instances increase in size. The reason for this behavior is the fact that the column generation algorithms require several iterations to converge and, in the small instances, this additional time is not sufficiently compensated by the faster convergence.

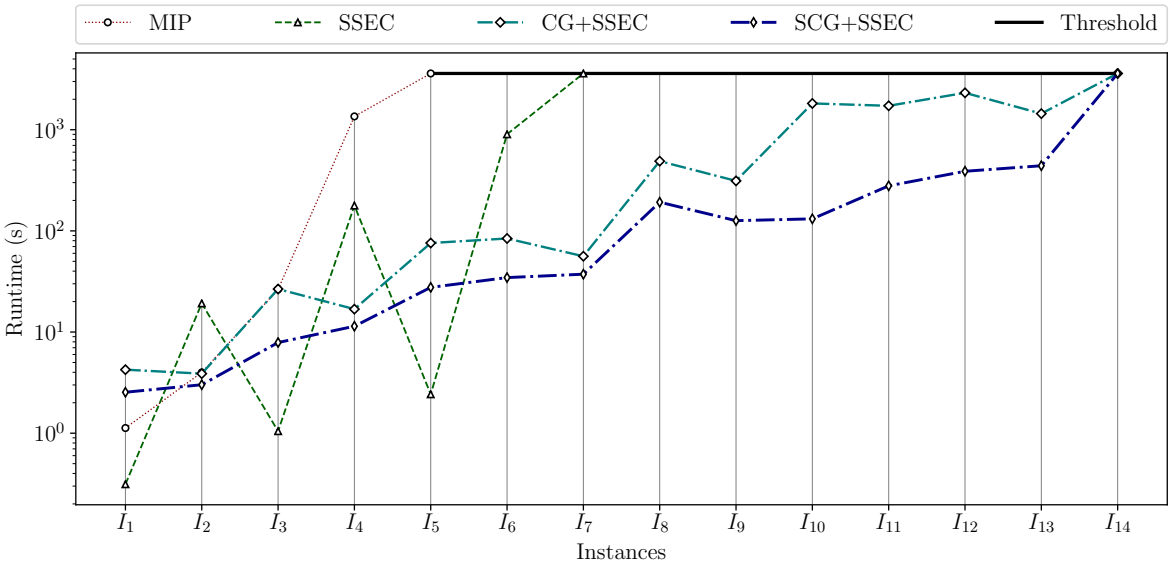


Figure 3: Runtimes of the different instances

5.3. Influence of instance parameters on runtime

In this section, the influence of the different instance parameters on the runtime is discussed. Several experiments are conducted in which four to five instances are solved with the original MIP and the two improvements. Each instance has the same set of parameters except for the parameter in question, which varies from instance to instance in order to isolate the influence of said parameter. The "base instance" I_{base} has 2 buses, 4 mandatory stops, 3 optional stops per cluster, 3 clusters, 16 requests and a capacity of 15 passengers per bus. The remaining instance parameters are the same as in the other instances that were previously discussed. This instance is modified to create six sets of instances. In the first five sets, either the number of passenger requests, the number of buses, the bus capacity, the number of optional stops per cluster, or the maximum walking time D^{w} varies while the other instance parameters are constant. In set six, the weights of the objective function vary. The choice of I_{base} is arbitrary since we are only interested in the relative increase or decrease of runtime. Again, in case the optimal solution is not found after one hour, the best feasible solution up until that point in time is reported. In Table 5, the influence of the different instance parameters on the runtime is shown.

				MIP	*SSEC*	*CG + SSEC*	*SCG + SSEC*
Inst.	**Parameter**		**V**	**Runtime (s)**			
$I_{r,1}$	**|P|**	12	2340	24.74	0.314	6.512	3.414
$I_{r,2}$		16	3120	27.71	1.003	26.21	7.874
$I_{r,3}$		20	3900	32.13	1.041	52.67	21.78
$I_{r,4}$		26	5070	28.74	1.471	224.1	87.21
$I_{b,1}$	**|B|**	2	3120	26.14	1.124	26.74	7.874
$I_{b,2}$		3	3328	481.4	44.47	17.25	6.817
$I_{b,3}$		4	3536	3600	244.1	11.98	6.144
$I_{b,4}$		5	3744	3600	3600	10.21	5.487
$I_{b,5}$		6	3952	3600	3600	8.120	4.914
$I_{c,1}$	**C**	9	3120	27.81	0.987	4.012	1.457
$I_{c,2}$		12	3120	29.74	1.541	6.451	2.147
$I_{c,3}$		15	3120	26.78	1.004	26.14	7.874
$I_{c,4}$		20	3120	29.12	0.914	52.14	22.14
$I_{s,1}$	**K**	1	1008	0.415	0.312	3.745	1.784
$I_{s,2}$		2	1920	2.215	0.397	12.74	4.147
$I_{s,3}$		3	3120	28.58	1.074	26.41	7.874
$I_{s,4}$		4	4608	3600	25.14	93.46	27.89
$I_{s,5}$		5	6384	3600	3600	217.1	91.78
$I_{o,1}$	**W**	[0.25, 0.35, 0.40]	3120	27.83	1.052	26.41	7.874
$I_{o,2}$		[0.33, 0.33, 0.33]	3120	32.12	1.423	35.00	9.748
$I_{o,3}$		[1, 0, 0]	3120	35.23	2.591	21.84	6.147
$I_{o,4}$		[0, 1, 0]	3120	3.574	0.474	2.304	1.647
$I_{o,5}$		[0, 0, 1]	3120	22.60	1.352	46.00	15.47

Table 5: Influence of the instance parameters on the runtimes

It can be seen that the runtimes of the SSEC and column generation solution methods increase with the number of requests. The increase in runtime is, however, greater for the column generation improvements. It needs to be noted that, in these small instances, the SSEC solution method has lower runtimes than the column generation solution methods. This is explained by the low number of buses: the number of ways in which passengers can be assigned to buses is limited and this results is tight lower bounds for the relaxed model. Tight lower bounds result in a fast convergence for the SSEC, while the column generation methods still need several iterations to converge. Results, which are not displayed in this paper, show that when the number of buses for these instances is raised from two to three, the column generation methods become faster than SSEC in all instances.

Remarkably, the runtimes of the column generation solution method decrease with the number of buses even though the number of variables increases. This is likely due to the nature of the restated model, in which a specific number of buses needs to be chosen among a large number of buses. When more buses are used, the number of non-zero variables in the solution of the RMP is higher, which means that it is less likely to obtain degenerate solutions in the RMP that slow down the convergence of the CG algorithm. This decrease is evidently less pronounced for the stabilized version of the column generation method. The runtimes of the MIP and the SSEC increase steeply with the number of buses. It can be seen that the column generation methods are faster than the MIP and SSEC when 3 or more buses are utilized.

It is clear that the capacity does not have a significant impact on the runtime of MIP and SSEC. The column generation approaches are, however, affected by the capacity, higher capacities make the runtime increase. This can be explained by the fact that more capacity implies more possibilities to distribute the passengers among the buses and thus more options need to be explored. Furthermore, the SSEC model deals well with an increase in optional bus stops, and is faster than the CG algorithm for most of the instances. The SSEC was implemented to deal with SECs and thus to deal with an increase in bus stops.

Table 5 also shows the influence of the weights W_1, W_2 and W_3 on the runtime. The parameter **W** refers to the values of the weights as a vector $[W_1, W_2, W_3]$. The first instance of this set has the original semi-arbitrary set of weights, the second instance has equal weights, and the remaining sets exclusively optimize one of the three objective function components. It can be seen that the second instance is more time consuming to solve than the first. This is likely due to the fact that all three objective function components are given equal importance, making it harder to find the best compromise. For the column generation methods, the instance that solely optimizes arrival times is the most time consuming, while solely optimizing the travel time is more difficult in the other solution methods. This can be explained by the fact that difficulty of the column generation methods lies in the passenger-bus assignment, which is correlated with the arrival times of the passengers. The difficulty of the other methods lies mainly in the satisfaction of the SEC, which are correlated with the travel times. For all instances optimizing the walking time of the passengers is the least time consuming. This is easy to understand, since in this case the problem is simplified to determining the closest available bus stop for each passenger.

Evidently, the largest influencing factors for the runtime of the column generation methods are the number of buses and the number of requests. More requests slow down the column generation algorithms while more buses increase its speed. For the MIP and the SSEC, the number of stops and the number of buses are the biggest contributors to an increase of runtime. On the other hand, the capacity and the number of requests do not have an effect on MIP and SSEC but do have an influence on the runtimes of column generation methods.

5.4. Qualitative analysis

In this section, a qualitative analysis is presented. First, a single solution is analyzed in detail. Second, the service quality is discussed. Lastly, the influence of different instance parameters is studied.

5.4.1. Analysis of a single solution

To illustrate what a solution of the DRFS looks like, the optimal solution of instance I_2 is discussed. The routing of the buses, the assignment of passengers to buses and departure bus stops are shown in Figure 4. The solid lines represent the routing of the buses. Each bus has a different route and can be distinguished by the color of the line. As previously mentioned, each bus starts in m_0, i.e., the first mandatory stop, and ends in m_2, i.e., the last mandatory stop. Passengers are denoted by a p followed by a number, for example, passenger 1 is denoted as p_1. The passengers walk from their origin location to their departure bus stop, which is displayed as a dotted line. The color of the dotted line of a passenger has the same color as the bus the passenger is boarding.

The timetable for the solution of this instance is shown in Table 6. The departure time D (in minutes), relative to an arbitrary reference point, of each bus is shown in the second column. The passengers that are picked up at a certain bus stop are shown in column PU.

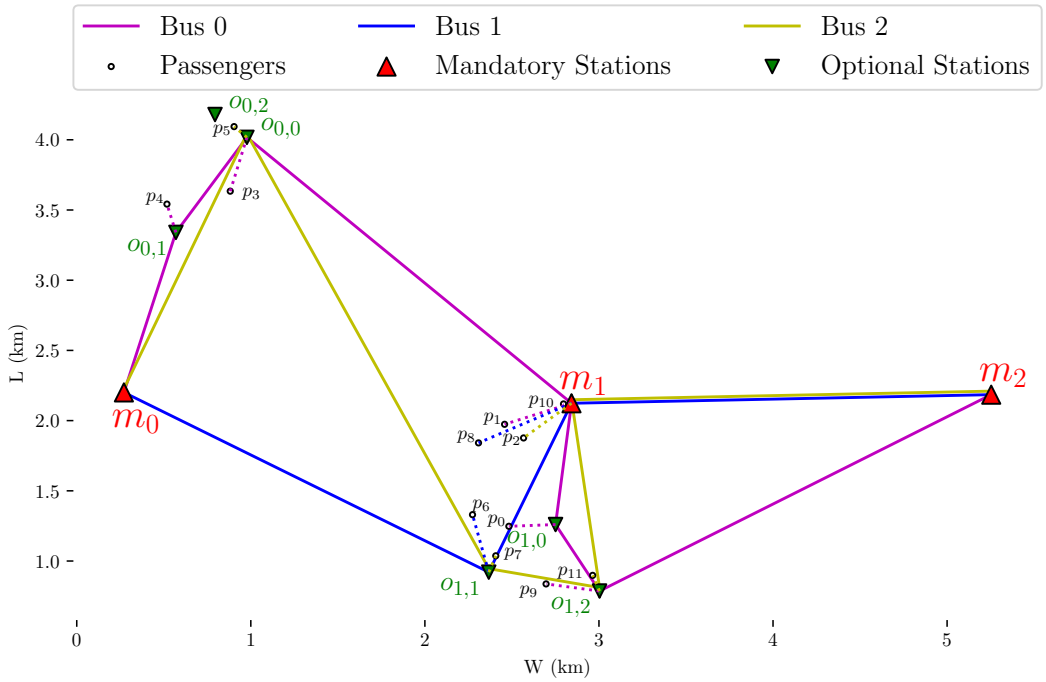


Figure 4: Routes and assignment of passengers to buses and departure bus stops

Bus 0			*Bus 1*			*Bus 2*		
Stop	**D(min)**	**PU**	**Stop**	**D(min)**	**PU**	**Stop**	**D(min)**	**PU**
m_0	2		m_0	0		m_0	11	
$o_{1,1}$	7	p_6	$o_{0,1}$	3	p_4	$o_{0,0}$	16	p_5
m_1	11	p_8	$o_{0,0}$	5	p_3	$o_{1,1}$	23	p_7
m_2	15		m_1	11	p_1, p_{10}	$o_{1,2}$	25	p_{11}
			$o_{1,0}$	13	p_0	m_1	28	p_2
			$o_{1,2}$	15	p_9	m_2	32	
			m_2	21				

Table 6: Timetable and passenger pick ups

5.4.2. Service quality

The service quality of the passengers in the DRFS is measured as a weighted sum of three components: the walking time to the departure stop, the difference between the actual arrival time and the desired arrival time, and the on-board travel time. In this optimization model, the travel time of the buses is optimized rather than the travel time of the passengers. One of the main reasons to do this is because the mathematical model becomes much more complex if the on-board travel times are modeled, and this leads to much higher runtimes. For example, instance I_1 takes over an hour to solve to optimality when the travel times of the passengers are optimized, compared to a few seconds with the original model. Furthermore, optimizing the bus routes also implies shorter travel times for passengers onboard. In this problem specifically, part of the bus route is fixed, meaning that the optimal bus route will not differ by much from the optimal route for the passengers. To illustrate this, we optimize the first four instances with a different objective function that optimizes the on-board travel time of the passengers and compare this to the original model. We denote the original model as Model 1 and the new model as Model 2. These results can be seen in Table 7, where the travel times **T**, the walking times **W** and the difference of arrival times **AD** can be seen for both models. Table 7 also shows the objective function value **Obj.**, calculated

according to Model 2, that would result from the optimal solutions of both models. Δ_1 denotes the average difference in onboard time per passenger between both models. Δ_2 denotes the relative difference of **Obj.** between both models.

	Model 1				*Model 2*					
	T	**W**	**AD**	**Obj.**	**T**	**W**	**AD**	**Obj.**	Δ_1	Δ_2
I_1	6244	4311	2648	4129	6244	4311	2648	4129	0	0.0%
I_2	6954	4256	1569	3856	5581	4301	1866	3647	114	5.4%
I_3	15270	6044	4736	7827	14335	6153	4736	7632	58	2.5%
I_4	14079	5989	2911	6781	11899	6153	3176	6399	136	5.6%

Table 7: Service quality of model 1 and model 2

Clearly, the difference in both the on-board travel times and the overall objective function is limited. On average, the passenger on-board time is at most 136 seconds longer with the original model, compared to Model 2. The objective function value improves 5.6% at most, when the travel times of the passengers are optimized, and in the first instance, there is no improvement at all. Optimizing the travel times of the buses rather than the travel times of the passengers can be justified by a limited decrease in service quality in exchange for a large increase in computational speed. Furthermore, the onboard times are typically less important to customers compared to the reliability of the service and arrival time at their destination [6, 7].

To further analyze the service quality of the optimal solutions of Model 1, we consider the User Journey Time (UJT) as a metric. The UJT is the sum of the travel times of the passengers and the walking time to their departure bus stop. We compare the UJT that is obtained by solving Model 1, with the UJT of a best case scenario. The latter is the case where all passengers are assigned to an individual bus, meaning that they always walk to their closest bus stop and are transported directly to the destination, similar to a taxi service. This is the lowest possible UJT, but it would obviously be a very costly operation. Table 8 shows the results for instance I_1 to I_{14}, where $\mathbf{\Delta_{UJT}}$ is the relative difference of UJT between the DRFS and the best case scenario.

It can be seen that the difference in UJT fluctuates between 14% and 31%. Instances where there are more buses available compared to the number of requests have a lower Δ_{UJT}. This is to be expected since, in these instances, there are more possibilities to offer each passenger a more customized service. The biggest difference in UJT comes from the difference in travel times, rather than the walking time. This is because, in these instances, passengers are often already assigned to their closest bus stop. Overall, it can be concluded that the DRFS offers a good service, in which the UJT of the passengers is relatively close to the lowest possible UJT.

5.4.3. Comparison to other services

Galarza et al. [14] make a comparison between the service quality of the DRFS and a fully flexible transportation service with optional stops, and a traditional feeder service with fixed routes and timetables. On one hand, it was found that the traditional feeder service performed 60% worse on average in comparison to the DRFS. On the other hand, the fully flexible service performed 6% better than the DRFS, which is a small decrease in service quality considering that the fully flexible service is does not serve passengers without a reservation.

	DRFS			*Best case scenario*			
	T	**W**	**UJT**	**T**	**W**	**UJT**	$\mathbf{\Delta_{UJT}}$
I_1	6244	4311	10555	4821	4256	9077	14%
I_2	6954	4256	11210	4861	4256	9117	19%
I_3	15270	6044	21315	9938	5989	15928	25%
I_4	14079	5989	20068	9938	5989	15928	21%
I_5	23773	8756	32529	16176	8701	24877	24%
I_6	22572	8701	31272	15900	8701	24601	21%
I_7	19840	8756	28596	15850	8701	24551	14%
I_8	38262	10822	49085	23797	10767	34564	30%
I_9	33753	10822	44575	23797	10767	34564	22%
I_{10}	44362	12768	57130	26700	12713	39413	31%
I_{11}	37101	12768	49868	26644	12713	39357	21%
I_{12}	44047	12768	56815	33591	12713	46303	19%
I_{13}	43982	12768	56750	33581	12713	46293	18%
I_{14}	80557	18428	98985	62675	18247	80923	18%

Table 8: User journey time of model 1 and the best possible solution

We further analyze the service quality by comparing the DRFS to a service that offers a door-to-door service to passengers that make a request for transportation. Similarly to a MAST service, mandatory stops are still serviced by each bus, but the bus can deviate from the main route to service passengers at their doorstep. The main difference with MAST is that, in MAST, the mandatory stops have fixed schedules. This service is the result of modifying the DRFS such that it offers a door-to-door service to passengers with a reservation rather than assigning them to a bus stop. We compare this service with the DRFS by testing both services on five different instances. Each instance has two buses with a capacity of 15 passengers per bus, four mandatory stops and three optional stops per cluster (not relevant for the door-to-door service) and a variable number of passenger requests. Furthermore, each instance has two sets of optional stops; set one is chosen such that passengers do not need to walk longer than 20 minutes to a bus stop and set two is chosen such that passengers walk at most 10 minutes to a bus stop. The number of requests and the location of the bus stops vary in order to observe the effect on the journey times and the onboard travel times. Other instance parameters have little to no effect on the comparisons and are thus omitted from the analysis. The results are given in Table 9, where $\mathbf{\Delta_T}$ is the relative difference in on-board travel time of the passengers and $\mathbf{\Delta_{UJT}}$ the relative difference of the total user journey time. The sub-index given to each performance metric denotes the set of optional stops.

		DRFS				*Door-to-door*				
	R	$\mathbf{T_1}$	$\mathbf{T_2}$	$\mathbf{W_1}$	$\mathbf{W_2}$	**T**	$\mathbf{\Delta_{T_1}}$	$\mathbf{\Delta_{T_2}}$	$\mathbf{\Delta_{UJT_1}}$	$\mathbf{\Delta_{UJT_2}}$
$I_{d,1}$	10	8164	7893	4109	2109	9507	-16.44%	-20.44%	22.54%	4.95%
$I_{d,2}$	15	11900	12562	5363	2704	14877	-25.02%	-18.43%	13.82%	2.55%
$I_{d,3}$	20	15476	15476	8822	4411	21128	-36.52%	-36.52%	13.05%	-6.24%
$I_{d,4}$	25	18697	19390	12548	6287	27052	-44.69%	-39.51%	13.42%	-5.35%
$I_{d,5}$	30	22500	22504	15787	7902	33212	-47.61%	-47.58%	13.25%	-9.23%

Table 9: User journey time of DRFS and a door-to-door service

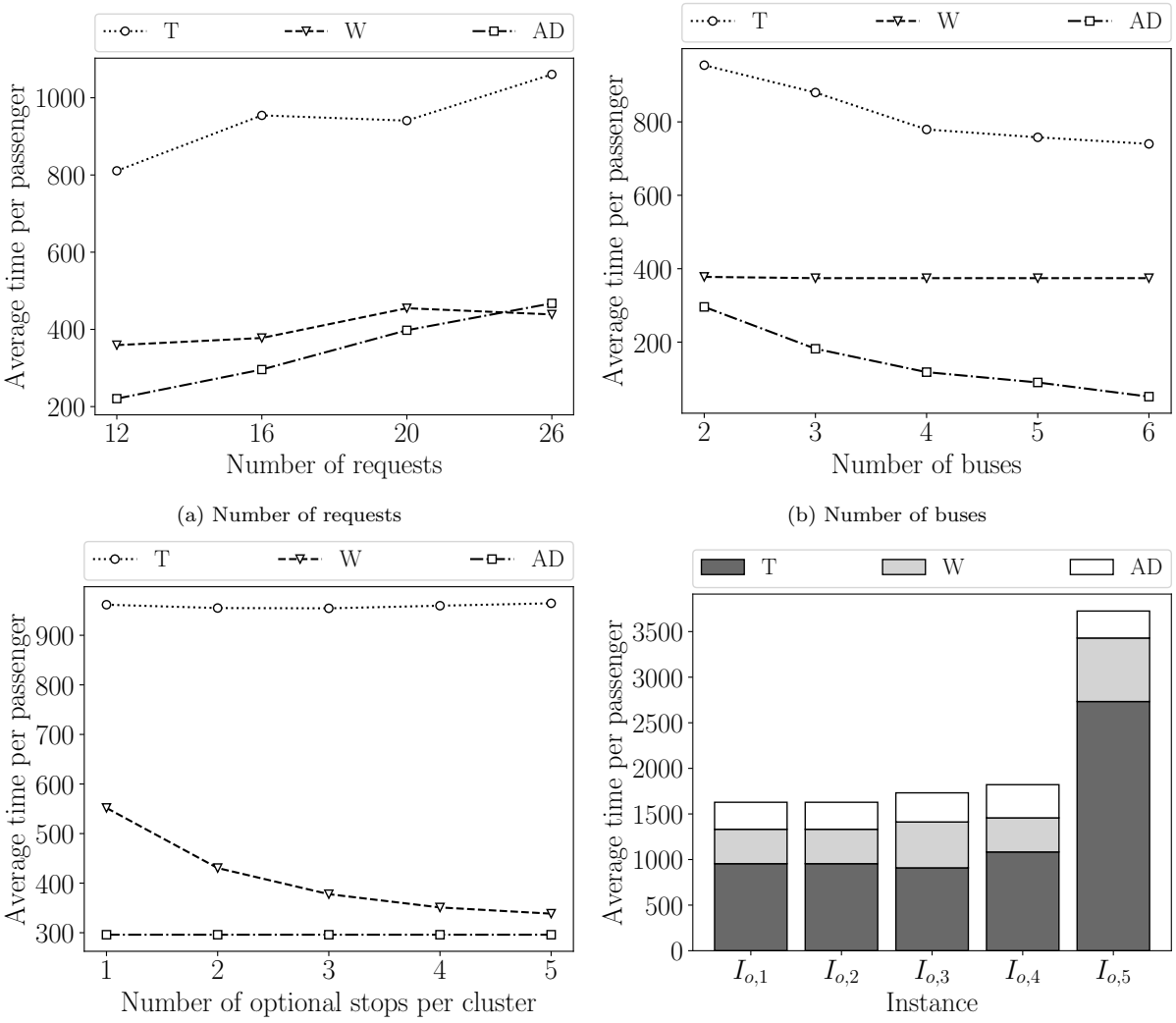
As expected, the on-board travel time of the passengers decreases, up to almost 50%, when

optional stops are used rather than a door-to-door service. This difference becomes larger when there are more passenger requests. The total journey time of the passengers is shorter for the door-to-door service when the walking time is longer for the DRFS or when there are relatively fewer requests. As more requests need to be served, however, this difference decreases. In set two, where the walking time is more limited due to a better placement of the stops, the user journey time becomes shorter for the DRFS. Although in most cases the total journey time for a door-to-door service is shorter on average, the onboard travel time of the passengers is always, and often significantly, shorter for the DRFS. Furthermore, longer onboard travel times infer longer time intervals between bus arrivals at the mandatory stops because each bus takes longer to pick up passengers in between the visits at the mandatory stops. This means that passengers without a reservation have longer waiting times at the mandatory stops in the door-to-door service.

5.4.4. Influence of instance parameters

The influence of instance parameters on the different service quality metrics is analyzed in this section. The quality metrics are the travel times (T), the walking times (W) and the difference in desired and actual arrival times (AD). This is done by measuring these metrics on the optimal solutions of different instances. In these instances, one parameter varies and the others remain constant. This methodology ensures the influence of each parameter is isolated. For this analysis, we use the same instances that are used in section 5.3. The number of buses, requests and optional stops per clusters as well as the objective function weights W_1, W_2 and W_3 are considered as varying parameters. The objective function weights are tested in five cases: with the original values ($I_{c,1}$), with equal values ($I_{c,2}$), when only T is optimized ($I_{c,3}$), when only W is optimized ($I_{c,4}$) and when only AD is optimized ($I_{c,5}$). The influence of the bus capacity and the maximum walking time is nonexistent on these smaller instance and is thus omitted from this analysis. Figure 5 shows the evolution of T, W and AD in function of the different instance parameters.

Evidently, an increasing number of requests also increases T, W and AD. This is to be expected, more requests means that the buses need to pick up more passengers at a time. This limits the customizability of the passengers' journeys and increases the travel times of the buses. The walking time increases less steeply than the other metrics. This is likely because, at a certain point, enough passengers share the same optimal departure stop, making it unnecessary to make certain passengers walk more. Similarly, having more available buses decreases T and AD significantly because the journeys of the passengers can be more customized to their specific needs. The walking time only decreases slight when more buses are used. This is because even with the smallest number of buses, most of the passengers are already assigned to their closest bus stop. On the one hand, walking times decrease significantly with the number of optional stops because this increases the probability of having available bus stops close to the passengers' location. On the other hand, the number of optional stops per cluster has a limited impact on T and AD. The travel time does not increase by much because the optional stops are scattered across the same area, so visiting more bus stops in the same clusters does not increase the driving time much either. The number of optional stops has no impact on AD because there is no correlation between the two. The influence of the objective function weights on T, W and AD is quite straightforward; when a certain metric is given more priority in the objective function, it will be optimized more. However, it should be noted that only optimizing AD has significantly large negative effects on T, and to a lesser extend on W as well. This is likely because in this scenario, the algorithm will try to make the buses arrive at the destination at very specific times, without any regard towards the efficiency of the routing. Furthermore, it can be seen that T usually has a higher value than W, which in turn is usually larger than AD.



(a) Number of requests

(b) Number of buses

(c) Optional stops per cluster

(d) Objective function weights

Figure 5: Influence of different instance parameters

6. Conclusion

In this paper, a feeder service with both optional and mandatory bus stops, presented in [14], is optimized with three different exact approaches to optimally plan the operation of the buses in this service. The service is demand-responsive since it selects which of the clustered optional bus stops to visit, based on passenger requests. On the other hand, there is predictability in the mandatory bus stops, which are always visited by each bus. While passengers can have a customized service by making requests, passengers that do not make such requests can still catch a bus at a mandatory stop. The service incorporates positive characteristics from both traditional transport services and on-demand transport services, with the aim of improving service quality.

In order to plan the operation of the buses in such a feeder service, a mixed integer programming model is developed. Solving this mathematical model with CPLEX, however, results in large runtimes for non-trivial instances of the problem. To reduce the runtimes, two different techniques are proposed: a separation of cycle elimination constraints and a column generation algorithm. Separation allows the model to find good, feasible solutions in a short amount of time, in comparison to the original mixed integer programming model. However, in some cases, the algorithm does

not converge quickly, due to bad lower bounds, leading to high integrality gaps. To redirect the gap difficulty towards a number of variable difficulty, the column generation procedure is developed, which requires a reformulation of the problem. The separation algorithm is still used in the subproblem to increase the speed of the algorithm. The column generation model yields integer optimal solutions, which means that solutions are optimal for the original model. This also means that there is no need to embed the column generation algorithm in a branch-and-price algorithm to obtain optimal solutions. All models are tested on a set of instances, with varying sizes. It is found that the column generation model outperforms the other models in most cases, and is able to successfully solve larger instances.

The main contribution of this paper is to provide an exact optimization method for a recently introduced feeder service with mandatory and clustered optional stops [14]. With techniques like separation and column generation, mid-sized instances of this problem can now be solved within reasonable time. This paper limits itself to exact methods in order to provide optimal solutions for available benchmark instances in order to evaluate the meta-heuristic presented in [14]. Further research will focus on developing algorithms to solve this problem in a dynamic environment, where not all requests are known during the planning phase. Another focus point for further research is to quantify the service quality of the passengers that do not make a request for transportation but still need it. The demand for transportation of these passengers is much more stochastic in nature and thus deserves more attention. Furthermore, future research should focus on solving some of the shortcomings of the DRFS, such as ensuring regularity of buses passing by the mandatory stops. Introducing a maximum headway at the mandatory stops will guarantee a maximum waiting time for passengers that did not make a formal request and are waiting for a bus at the mandatory stops. This addition makes the optimization process much more complex and thus requires more research on the modeling and the solution methods that are needed to optimize the system.

References

[1] M. Steriu, Statistics brief - Local public transport trends in the European Union (2016). URL `https://www.uitp.org/statistics-brief-public-transport-in-the-EU`

[2] J. Hine, F. Mitchell, Better for everyone? Travel experiences and transport exclusion, Urban Studies 38 (2) (2001) 319–332.

[3] J. Anable, 'Complacent Car Addicts'; or 'Aspiring Environmentalists'? Identifying travel behaviour segments using attitude theory, Transport Policy 12 (1) (2005) 65–78.

[4] S. Handy, L. Weston, P. L. Mokhtarian, Driving by choice or necessity?, Transportation Research Part A: Policy and Practice 39 (2005) 183–203.

[5] G. Beirão, J. A. Sarsfield Cabral, Understanding attitudes towards public transport and private car: A qualitative study, Transport Policy 14 (6) (2007) 478–489.

[6] L. Dell'Olio, A. Ibeas, P. Cecin, The quality of service desired by public transport users, Transport Policy 18 (1) (2011) 217–227.

[7] D. A. Hensher, P. Stopher, P. Bullock, Service quality - developing a service quality index in the provision of commercial bus contracts, Transportation Research Part A: Policy and Practice 37 (6) (2003) 499–517.

[8] M. Friman, B. Edvardsson, T. Gärling, Frequency of negative critical incidents and satisfaction with public transport services. I, Journal of Retailing and Consumer Services 8 (2) (2001) 95–104.

[9] D. van den Buuse, A. Kolk, An exploration of smart city approaches by international ICT firms, Technological Forecasting and Social Change 142 (2019) 220–234.

[10] A. Ceder, N. H. M. Wilson, Bus Network Design, Transportation Research Part B: Methodological 208 (4) (1986) 331–344.

[11] X. Li, L. Quadrifoglio, Feeder transit services: Choosing between fixed and demand responsive policy, Transportation Research Part C: Emerging Technologies 18 (5) (2010) 770–780.

[12] J. D. Nelson, S. Wright, B. Masson, G. Ambrosino, A. Naniopoulos, Recent developments in Flexible Transport Services, Research in Transportation Economics 29 (1) (2010) 243–248.

[13] M. J. Alonso-González, T. Liu, O. Cats, N. Van Oort, S. Hoogendoorn, The Potential of Demand-Responsive Transport as a Complement to Public Transport: An Assessment Framework and an Empirical Evaluation, Transportation Research Record 2672 (8) (2018) 879–889.

[14] B. D. Galarza Montenegro, K. Sörensen, P. Vansteenwegen, A large neighborhood search algorithm to optimize a demand-responsive feeder service, Transportation Research Part C: Emerging Technologies 127 (2021) 103102.

[15] A. Ceder, Introduction into transit service planning, in: Public transit planning and operation: modeling, practice and behavior, 2nd Edition, CRC Press, Boca Ranton, 2016, Ch. 1, pp. 1–21.

[16] R. M. Lusby, J. Larsen, S. Bull, A survey on robustness in railway planning, European Journal of Operational Research 266 (1) (2018) 1–15.

[17] A. Schöbel, Line planning in public transportation: Models and methods, OR Spectrum 34 (3) (2012) 491–510.

[18] C. Iliopoulou, K. Kepaptsoglou, E. Vlahogianni, Metaheuristics for the transit route network design problem: a review and comparative analysis, Public Transport 11 (3) (2019) 487–512.

[19] S. Carosi, A. Frangioni, L. Galli, L. Girardi, G. Vallese, A matheuristic for integrated timetabling and vehicle scheduling, Transportation Research Part B: Methodological 127 (2019) 99–124.

[20] A. Schöbel, An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation, Transportation Research Part C: Emerging Technologies 74 (2017) 348–365.

[21] N. Agatz, A. Erera, M. Savelsbergh, X. Wang, Optimization for dynamic ride-sharing: A review, European Journal of Operational Research 223 (2) (2012) 295–303.

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

[23] W. A. Ellegood, S. Solomon, J. North, J. F. Campbell, School bus routing problem: Contemporary trends and research directions, Omega (United Kingdom) 95 (2020) 102056.

[24] C. Archetti, M. Speranza, D. Weyland, A simulation study of an on-demand transportation system, International Transactions in Operations Research 25 (4) (2018) 1137–1161.

[25] J. F. Potts, M. A. Marshall, E. C. Crockett, J. Washington, Best Practices of Successful Flexible Public Transportation Services, in: Transit Cooperative Research Program (TCRP) Report 140: A Guide for Planning and Operating Flexible Public Transportation Services, Transportation Research Board, Washington, D.C., 2010, Ch. 4, pp. 42–88.

[26] L. Quadrifoglio, M. M. Dessouky, F. Ordóñez, Mobility allowance shuttle transit (MAST) services: MIP formulation and strengthening with logic constraints, European Journal of Operational Research 185 (2) (2008) 481–494.

[27] J. Zhao, M. Dessouky, Service capacity design problems for mobility allowance shuttle transit systems, Transportation Research Part B: Methodological 42 (2) (2008) 135–146.

[28] T. Liu, A. Ceder, Analysis of a new public-transport-service concept: Customized bus in China, Transport Policy 39 (2015) (2015) 63–76.

[29] M. Kim, P. Schonfeld, Conventional, flexible and variable-type bus services, Journal of Transportation Engineering (2012) 263–273.

[30] L. Fu, Q. Liu, Real-Time Optimization Model for Dynamic Scheduling of Transit Operations, Transportation Research Record 1 (1857) (2003) 48–55.

[31] C. L. Martins, M. V. Pato, Search strategies for the feeder bus network design problem, European Journal of Operational Research 106 (1998) 425–440.

[32] F. Ciaffi, E. Cipriani, M. Petrelli, Feeder Bus Network Design Problem: a New Metaheuristic Procedure and Real Size Applications, Procedia - Social and Behavioral Sciences 54 (2012) 798–807.

[33] J. J. Lin, H. I. Wong, Optimization of a feeder-bus route design by using a multiobjective programming approach, Transportation Planning and Technology 37 (5) (2014) 430–449.

[34] M. Mistretta, J. A. Goodwill, R. Gregg, C. DeAnnuntis, Best Practices in Transit Service Planning. Final Report No. BD549-38, Tech. rep., Center for Urban Transportation Research for the Florida Department of Transportation (2009).

[35] B. Sun, M. Wei, S. Zhu, Optimal design of demand-responsive feeder transit services with passengers' multiple time windows and satisfaction, Future Internet 10 (3) (2018).

[36] X. Lu, J. Yu, X. Yang, S. Pan, N. Zou, Flexible feeder transit route design to enhance service accessibility in urban area, Journal of Advanced Transportation 50 (4) (2015) 507–521.

[37] F. Qiu, W. Li, A. Haghani, An exploration of the demand limit for flex-route as feeder transit services: a case study in Salt Lake City, Public Transport 7 (2) (2015) 259–276.

[38] M. Stiglic, N. Agatz, M. Savelsbergh, M. Gradisar, Enhancing urban mobility: Integrating ride-sharing and public transit, Computers and Operations Research 90 (2018) 12–21.

[39] A. U. Raghunathan, D. Bergman, J. Hooker, T. Serra, S. Kobori, The Integrated Last-Mile Transportation Problem (ILMTP), Proceedings International Conference on Automated Planning and Scheduling, ICAPS 2018-June (2018) 388–397.

[40] N. Marković, M. E. Kim, E. Kim, S. Milinković, A threshold policy for dispatching vehicles in demand-responsive transit systems, Promet - Traffic - Traffico 31 (4) (2019) 387–395.

[41] DeLijn, DIAL-A-BUS A SOLUTION FOR YOUR JOURNEY? (2021).
URL `https://www.delijn.be/en/belbus/?vertaling=true`

[42] M. Wei, T. Liu, B. Sun, B. Jing, Optimal Integrated Model for Feeder Transit Route Design and Frequency-Setting Problem with Stop Selection, Journal of Advanced Transportation 2020 (2020) 1–12.

[43] L. Wosley, Integer Programming, John Wiley & Sons Inc, New York, United States, 1998.

[44] M. Fischetti, P. Toth, A polyhedral approach to the asymmetric traveling salesman problem, Management Science 43 (11) (1997) 1520–1536.

[45] D. M. Ryan, B. A. Foster, An integer programming approach to scheduling, Computer scheduling of public transport urban passenger vehicle and crew scheduling (1981) 269–280.

[46] F. Vanderbeck, L. A. Wolsey, An exact algorithm for ip column generation, Operations Research Letters 19 (4) (1996) 151–159.

[47] Y. Dodge, An introduction to L1-norm based statistical data analysis, Computational Statistics and Data Analysis 5 (4) (1987) 239–253.

[48] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen, Stabilized column generation, Discrete Mathematics 194 (1) (1999) 229–237.