Received: 6 September 2022 | Revised: 13 June 2023 | Accepted: 17 August 2023

DOI: 10.1002/net.22185

RESEARCH ARTICLE

WILEY

A demand-responsive feeder service with a maximum headway at mandatory stops

Bryan David Galarza Montenegro[1] | **Kenneth Sörensen**[1] | **Pieter Vansteenwegen**[2]

[1]Department of Engineering Management (ENM), University of Antwerp, Antwerp, Belgium
[2]KU Leuven Institute for Mobility - CIB, KU Leuven, Leuven, Belgium

Correspondence
Bryan David Galarza Montenegro, Department of Engineering Management (ENM), University of Antwerp, Prinsstraat 13, 2000, Antwerp, Belgium.
Email: bryan.galarzamontenegro@uantwerpen.be

Funding information
Fonds Wetenschappelijk Onderzoek, Grant/Award Number: G.0759.19N

Abstract

Public transportation out of suburban or rural areas is crucial. Feeder transportation services offer a solution by transporting passengers to areas where more options for public transport are available. On one hand, fully flexible demand-responsive feeder services (DRFSs) efficiently tailor their service to the needs of the passengers. On the other hand, traditional feeder services provide predictability and easier cost control. In this article, a semi-flexible DRFS is considered, which combines positive characteristics of both traditional services as well as fully flexible services. This feeder service has two types of bus stops: mandatory bus stops and optional bus stops. Mandatory bus stops are guaranteed to be visited by a bus within a certain time interval. Optional stops are only visited when there is demand for transportation nearby. The performance of this feeder service is optimized with the use of a new type of metaheuristic framework, which we denote as *parameter space search*. Experimental results on small benchmark instances indicate that the heuristic performs on average 12.42% better than LocalSolver, a commercial optimization solver, with an average runtime of 2.1 s. Larger instances can also be solved, typically within 2 min.

KEYWORDS

demand-responsive transportation, feeder service, meta-heuristics, public bus transport

1 | INTRODUCTION

In 2015, 243 billion public transport journeys were made around the world, an 18% increase over the previous 15 years [49]. Mobility is essential for the development of societies since it enables accessibility to social activities, goods and services. An efficient and adequate public transport service is thus crucial in reducing social exclusion and poverty [1, 27]. Furthermore, public transportation can have a positive impact in the urbanization and infrastructure planning of cities [1]. Although the use of public transport has increased over the years, there is still a preference for private transport over the use of public transport [26]. This has led to increased pollution, congestion and overfull parking lots. In order to make public transport a more attractive option, the service quality of transportation services must increase. For secluded areas in particular, like residential areas and suburbs, a feeder service enables connectivity to major transit networks. Passengers from typically sparsely populated areas are transported to areas with a high demand for transportation or to transportation hubs, where they can continue their journey. These services can be an answer to the problem of overfull parking lots at transportation hubs and congestion in the surrounding area.

In a feeder service, all passengers have the same destination but different origins. The bus line in the feeder service with mandatory stops (FSMS) considered in this article serves two sets of predetermined bus stops: mandatory stops and clustered optional stops. The mandatory stops are visited by each bus in the bus line and have a maximum allowable headway, that is, these stops are visited by a bus within a certain time interval. The optional stops are only visited by a bus when a passenger nearby makes a request for transportation. Passengers make a request for transportation to the transportation hub, by stating

their current location and the time they wish to depart or their desired arrival time at the hub. Passengers are assigned to a departure bus stop, where they have to walk to. It is assumed that passenger requests are known before a certain deadline. After this deadline, the operation of the service for the planning horizon is determined and cannot be modified further. In other words, all passenger demand is assumed to be static. However, this means that the routes and the timetables of the buses are not completely fixed but can be modified according to the received requests.

The main contribution of this article is the design of a new semi-flexible demand-responsive feeder service (DRFS). The FSMS can be a good alternative to both traditional transportation services and fully flexible transportation services because it offers positive characteristics of both types of services. The mandatory stops are a good way to meet demand that did not made an explicit request for transportation and provide a sense of predictability for all passengers (both passengers with and without a reservation). Optional stops and online requests offer a more customized experience to passengers that have explicitly made a request. Furthermore, Beirão and Sarsfield Cabral [7] show that rather than waiting time, the uncertain arrival time of transport is of most importance, that is, the reliability of the service. Attributes such as comfort and arrival time at the destination are also highly valued by passengers. These are attributes that the FSMS is designed to improve upon, which means that the FSMS can be a viable alternative to private transport. This article also introduces the *parameter space search* (PSS) heuristic, a new type of metaheuristic concept. PSS executes an underlying constructive heuristic multiple times with different heuristic parameters. An overlying local search heuristic, we use simulated annealing (SA) for our problem, aims to find the best possible parameter values in the parameter space of the underlying constructive heuristic. The results obtained by the heuristic prove that good quality solutions are obtained within relatively small runtimes.

In the next section, a literature review on public transport feeder services is presented. In Section 3, we present a detailed description of the FSMS and a mathematical model to optimize the service. Section 4 presents the PSS heuristic, which is developed to solve the optimization problem. Section 5 analyses the performance of the PSS heuristic in order to fine-tune its parameters. In Section 6, the results for several instances, obtained by solving the problem using the PSS heuristic, are presented and discussed. In Section 7, the influence of different instance parameters on the service quality is analyzed and discussed. The last section concludes the article and discusses plans for future research.

2 | LITERATURE REVIEW

The FSMS that is presented in this article is an extension of the DRFS presented in Galarza Montenegro et al. [22] and Galarza Montenegro et al. [23]. In the DRFS, a fleet of buses transports passengers from a suburban area to an area with high demand for transportation, such as a train station. The DRFS receives requests from passengers until a certain deadline before the first bus is dispatched. Each request states the passenger's location and desired arrival time at the destination. There are two sets of bus stops. Mandatory stops are always visited by each bus, while optional stops are visited if there is a demand for transportation nearby. Galarza Montenegro et al. [22] solve larger instances with a large neighborhood search heuristic, while Galarza Montenegro et al. [23] solve smaller instances with an exact method. The main contributions of the FSMS, with respect to the DRFS in Galarza Montenegro et al. [22] and Galarza Montenegro et al. [23], are as follows: the service now needs to guarantee that a bus departs from each mandatory stop within a certain time interval, the return trip of the buses is now explicitly considered and passengers can now state a desired departure time as well. These additions make the optimization of the service substantially more complex and improve the quality of the service. To the best of our knowledge, no other research has studied the FSMS presented in this article or the DRFS in Galarza Montenegro et al. [22] and Galarza Montenegro et al. [23]. Nevertheless, closely-related types of services have been considered in the literature and will be discussed in this section.

Traditional feeder services (TFS) have predetermined stops, routes, and timetables. In these TFS, the demand is considered to be known and is often derived from historical data. As such, these services result in feeder lines that perform well in terms of satisfaction of demand, and waiting time. The feeder bus network design problem (FBNDP) is often tackled for TFS. 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 [39]. Ciaffi et al. [14] solve the FBNDP using a heuristic algorithm to generate routes, together with a genetic algorithm (GA) to find the best possible network of routes and their frequencies. This solution approach is implemented for the networks of Winnipeg and Rome. Lin and Wong [35] present a multi-objective programming approach to solve the FBNDP. In this study, the 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. Mohaymany and Gholami [43] consider a FBNDP with multiple transportation modes, each mode with different capacities and performances. Ant colony optimization is used to solve the optimization model. Zheng et al. [59] introduce a "demand coefficient" to quantify the feeder demand and use a tabu search (TS) algorithm to optimize the design of the feeder network. The optimization approach is implemented in a downtown area of Suzhou, China. Often, the main objective while optimizing these services is to minimize travel times of passengers and transfer times to the main transit. Shrivastava and O'Mahony [51] use a GA combined with a specialized heuristic to optimize the routes

and the timetables of a TFS. In a different study, Shrivastava and O'Mahony [50] use a GA to optimize both the routes and the timetables simultaneously.

TFS have clear shortcomings, such as lack of accessibility and flexibility. In these services, it is difficult to accommodate the different needs of the passengers, such as desired arrival time or departure time. Furthermore, 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 [42]. These limitations have contributed to the rise of demand-responsive transportation services (DRTS). DRTS are bus services that do not operate using fixed routes and timetables, and take into consideration the individual demand for transportation. Consequently these services are better able to deal with sparse and ever-changing demand [2]. DRTS usually require passengers to make explicit requests for transportation in order to gain information about their needs. Some well-known DRTS are the dial-a-ride (DAR) problem [57], the mobility allowance shuttle transit service (MAST) [46], the demand-responsive connector (DRC) [33], and the customized bus (CB) [36]. The DAR service improves the accessibility of transit services by offering a door-to-door service to passengers that make a request with their desired pickup and drop-off locations. Sun et al. [52] present a mixed-integer linear programming model for a DRFS similar to a DAR problem. The model is solved using a heuristic algorithm and is used in a case study in Nanjing City, China. In a MAST service, vehicles have a fixed set of bus stops that they always need to visit, for example, 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. This concept has been applied to feeder services as well. Lu et al. [37] developed a three-stage heuristic algorithm to optimize such a problem, together with a bus assignment sub-problem. Furthermore, Qiu et al. [45] analyzed a feeder service similar to a MAST service, which has been implemented in Salt Lake city in the USA. In the DRC, no mandatory stops are considered and buses transport passengers from their origin location to transfer hubs within a predefined service area. Quadrifoglio and Li [47] and Li and Quadrifoglio [34] both present analytical and simulation models to help service providers choose between a fixed feeder service and a DRC, depending on operational circumstances. Passengers are able to notify their presence by means of a phone or Internet booking service. Immediately before the beginning of each trip, waiting customers are scheduled and the route for the trip in the service area is constructed. In the CB, an origin area is connected to a destination area with an express service and mostly dedicated lanes. However, the routes inside both areas are flexible. Guo et al. [25] develop a CB service, and use an exact model with time windows similar to the DAR problem. The solution includes intermediate stops where passengers can transfer to other lines and systems. The model is implemented in a case study in Beijing, China. The study compares the branch-and-cut results with a GA and a TS algorithm. A different type of service is presented in Crainic et al. [17] and Crainic et al. [16]. The service in both studies is a stop-based transportation system which they refer to as a class of demand-adaptive systems (DAS). In these systems, flexible routes are created for passengers that make a request for transportation, while the buses still serve mandatory stops at fixed schedules. At first, a master scheduling partially defines routes and time windows. Later, the actual schedule of each service is built to include optional stops. Requests can be rejected if they make the tour infeasible or unprofitable.

DRTS typically have more success in low demand areas with a sparse population, while TFS thrive in high-demand and densely populated areas. When and where to use which feeder service is further discussed by Li and Quadrifoglio [34]. The study shows that flexible feeder services perform better with lower demand rates and become progressively preferred when more importance is given to the walking time of the passengers.

Vansteenwegen et al. [53] present a recent and extensive survey and categorize DRTS into different groups. The transportation services are divided based on the degree of responsiveness; "static" when the planning is determined before a certain deadline and no changes are possible afterwards, and "dynamic" when the planning can be modified when new incoming requests are received. An example of static planning is the service in Lee and Savelsbergh [32]. The study considers a DRC to minimize operator costs, considering the time window for pickups and drop-offs. The authors consider the train frequency at the station and use these time windows as parameters of the operation. This allows the service providers to select the best operation period for servicing passengers. The authors use both a heuristic and an exact model to optimize the service before the start of operation, after which the planning is fixed. An example of dynamic planning is the service presented in Fu and Liu [21]. The authors implement a real-time scheduling model with dynamic stop skipping. The model optimizes the schedule of the vehicles just before departure from the depot. This makes modifications to the planning possible before the dispatching of the vehicles but not after. Vansteenwegen et al. [53] denote this responsiveness as "dynamic offline". More responsiveness can be found in the service presented by Pratelli et al. [44]. The service starts from a standard route and deviates from it when a request is made. Each route has a minimum and maximum number of deviating stops. Requests are received in real-time and each bus can change its planning at any time, even after the start of operation. This type of responsiveness is denoted as "dynamic online". The services are further distinguished by the level of flexibility. On one hand, "fully flexible" services have no fixed routes or timetables. Services like the DRC or a DAR service are fully flexible because there is no aspect that remains constant regardless of the demand. On the other hand, "semi-flexible" services have predetermined planning that can be modified in order to

meet the needs of the passengers. Services such as the MAST service or the customized bus are semi-flexible. For example the MAST service has a main route that all buses need to visit regardless of the demand, but buses can deviate from it to pick-up passengers. Our FSMS is classified as a static semi-flexible feeder service.

Vansteenwegen et al. [53] review 151 papers related to DRTS, out of which 43 papers study a feeder service. It was concluded that many feeder services operate using small vehicle capacities; many papers use a vehicle capacity of 10 passengers or less, and only a few use a capacity of 40 passengers or more. Almost half of the papers that study flexible feeder services do not consider a limited vehicle capacity at all. Moreover, excluding the DRFS, only three papers consider semi-flexible feeder services with a limited capacity [29-31]. In all three papers the capacity is 19 passengers or less.

Furthermore, the planning of flexible feeder services is often optimized with metaheuristic frameworks or other simple heuristics [53]. There are papers that utilize exact techniques, however, these methods are only used to find optimal solutions to be used as benchmarks for evaluating heuristic solution approaches. The only papers that solely use exact methods are Pratelli et al. [44], Chien et al. [13], Melachrinoudis et al. [41], Wang et al. [55], Zheng et al. [60], Lakatos et al. [31], and Mehran et al. [40]. In these cases, the instances are considered relatively small.

To solve the optimization problem of the FSMS, we introduce the PSS heuristic. In the PSS heuristic, an underlying constructive heuristic is executed multiple times with different construction parameters. An overlying local search heuristic operates on the parameter space of the constructive heuristic in order to find the construction parameter values that lead to the best solutions. The PSS heuristic is similar to a greedy randomized adaptive search procedure (GRASP) [48], where solutions are iteratively constructed with a greedy randomized algorithm and consequently improved. The difference between the PSS and GRASP is that, on one hand, the solutions constructed with GRASP are improved with local search in each iteration. On the other hand, the PSS heuristic uses local search to find the best construction parameters for the underlying construction algorithm. Unlike the GRASP, the PSS heuristic utilizes information of previous construction iterations to improve upon the solution. Therefore, the PSS heuristic as a whole can be viewed as an online automatic parameter fine-tuning procedure. This is similar to the F-race [8], the Iterative F-race procedure (I/F-race) [5, 9] and the Heuristic Oriented Racing Algorithm [6]. All of these methods automatically tune the parameters of optimization algorithms with the use of machine learning techniques, more specifically with racing approaches [38]. The difference is that the PSS heuristic uses local search rather than machine learning to automatically fine-tune the parameters of a greedy construction algorithm. Other parameter tuning procedures have utilized metaheuristics to fine-tune optimization algorithms as well. Population-based heuristics have been used to fine-tune optimization algorithms [3, 24, 58]. In such cases, individuals represent parameter configurations and an evaluation method for each configuration corresponds to the fitness function. Similarly, Hutter et al. [28] make use of iterated local search to solve the parameter tuning problem, while Cáceres and Stützle [12] employ variable neighborhood search. However, all of the mentioned parameter tuning procedures are a preliminary step that is taken once, before the optimization algorithm starts. Unlike these procedures, the parameter tuning in the PSS is an integral part of the optimization algorithm.

The optimization model of the FSMS shares some characteristics with the vehicle routing problem with time windows (VRPTW). In the more general vehicle routing problem (VRP), a set of customers, who are geographically dispersed, are served by a set of vehicles. The vehicles have a limited capacity and are dispatched from a central depot. The goal is to find the optimal routes to serve all customers with the available vehicles [15]. In the VRPTW, customers are only available during a certain time period, which imposes time window constraints for arrivals at the customers' location. Braekers et al. [11] review 277 papers that study a version of the VRP. It is concluded that the vast majority, namely 71.3%, of the papers optimize the VRP with the use of metaheuristics. Exact methods are utilized for 17.1% of the models. The remainder of the models are solved with other solution methods. It was also found that 90.5% of the papers consider a limited vehicle capacity and 37.9% study a VRPTW. Most of the VRPTW are solved with the use of heuristics. El-Sherbeny [20] and Dixit et al. [19] give an overview of the literature on VRPTW.

The vehicles in the VRPTW visit a node that is the location of a customer, while passengers cannot decide on the bus stop they are assigned to in the FSMS. The optimization algorithm of the FSMS, which takes into account a maximum walking duration, determines this. Furthermore, all nodes in the VRPTW are bound by a departure time window. This is not always the case with the FSMS, because some passenger requests specify a preferred arrival time at the destination, imposing only a time window constraint at the final node of the route. In the FSMS, the itinerary is also partially predetermined. The FSMS also has a maximum headway at mandatory stops, which influences vehicle scheduling and routing even more. This headway constraint also makes routes and timetables of different bus trips interdependent, which complicates the search for a feasible solution.

The FSMS resembles the feeder service variant of the MAST service and the DAS the most. Our feeder service has a fixed route where it can deviate from, just as in MAST and DAS services. The main difference is that MAST services provide a door-to-door service to some customers within a certain radius, while our service assigns and groups passengers at a limited number of bus stops. DAS work with bus stops, but there is no bus stop assignment involved and the walking times of the passengers are not considered. Bus stop assignment increases the efficiency of the bus assignment and the routing. The timetable for the fixed route is also preset and cannot be changed in MAST. In DAS, buses must depart from the stops in the fixed route

within certain time windows. This limits the time the buses in DAS or MAST can devote to deviating from the main route. Our service is more flexible, while still guaranteeing that at least one bus departs from mandatory bus stops within a certain time frame. Furthermore, the difficulty of satisfying the preset timetable at the mandatory stops in DAS and MAST is alleviated by allowing requests to be rejected. In our service, all requests must be accepted. This makes it more difficult to find feasible solutions. Finally, DAS and MAST are mostly optimized on small scale instances or the problem is divided into subproblems to reduce the complexity of the optimization model, which can lead to optimal solutions of lesser quality. Very few papers aim to solve MAST or DAS as an integrated problem as we will do in this article.

3 | PROBLEM DESCRIPTION

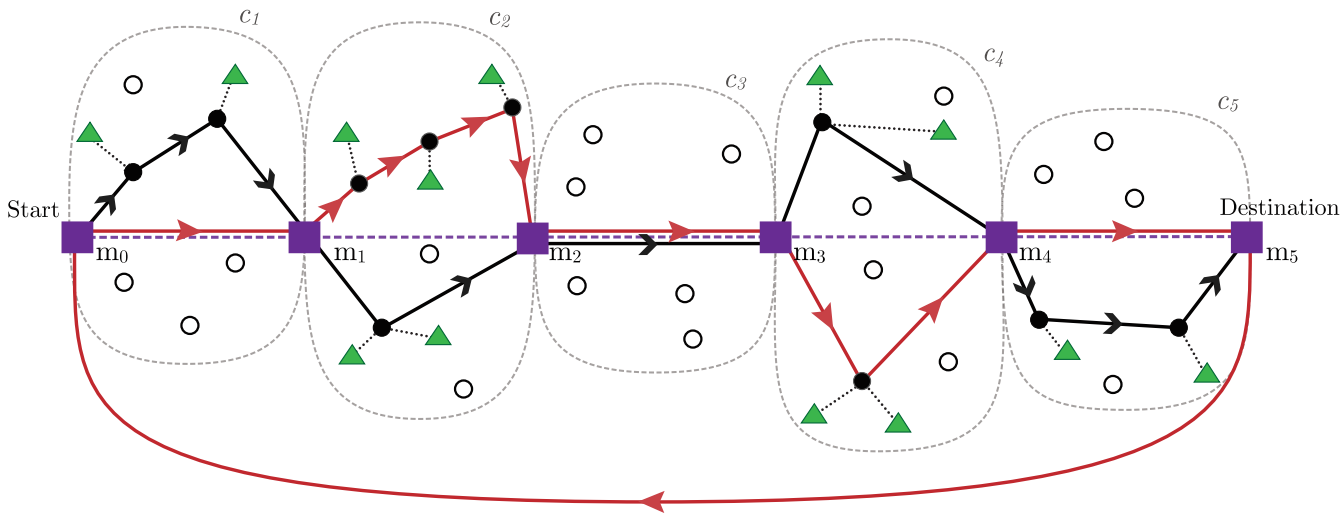
In this section, the FSMS is described in detail. First, the service is explained and the setting of the problem is determined. Next, a mathematical optimization model is defined in more formal terms.

3.1 | Description of the feeder service with mandatory stops

The FSMS is situated in an area with low demand for transportation, such as a suburban area. The bus lines of this service are designated shuttle buses that bring the inhabitants of this low demand area to a transportation hub or to a nearby city center, that is, the destination for all passengers is the same. In the rest of this article, a single bus line is considered because all bus lines in the FSMS are operated independently, assuming they each have their own area to cover and no bus stops are shared by different lines. The bus line is operated by a fleet of vehicles, that is, there are multiple buses with different schedules that travel along the same line. The buses serve two types of bus stops: mandatory stops and optional stops. Each bus visits all mandatory stops and these stops have a maximum allowable headway, that is, at least one bus must depart from each mandatory stop within a certain time interval after the previous bus. This means that passengers that did not make a formal request for transportation, waiting at mandatory stops do not need to wait longer than a certain amount of time for a bus. It should be noted that this is more strict than having a certain number of buses per hour. The optional stops are only visited when demand is assigned. Passengers are assigned to a departure stop that they have to walk to. This assignment takes into consideration a maximum walking time for each passenger. The mandatory stops can, for example, be placed along a 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. Each cluster is located between two mandatory stops. The buses always start at the first mandatory stop and end at the last mandatory stop, this is denoted as a "trip". A bus route can deviate from the route along the mandatory stops and 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 a next cluster. After a bus has reached the last mandatory stop, it goes back to the first mandatory stop following the shortest path without serving any stops. Afterwards, this bus can be reused for the next trip.

The service must serve a certain number of passengers during a planning horizon, which is typically a few hours. A passenger demands a ride online, through a website or a phone application, in which they state their starting location and their desired arrival time or their desired departure time. For a feeder service it might be more likely that passengers request a desired arrival time, however, there might also be passengers who prefer a desired departure time. Therefore, the additional option to request a desired departure time makes the service more complete. Passengers immediately get a reply that their request will be scheduled and that they will soon receive more details (departure stop and time), so they can start walking towards their departure stop well in time. The service receives requests until a certain deadline, for example, one or two hours before the first bus is dispatched. 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. After the planning is determined, the timetable of the buses at the mandatory and optional stops is known and shared publicly. 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. Passengers that are not aware of the request service or the timetable information can still catch a bus at the mandatory stops with a guaranteed (low) maximum waiting time. Passengers that did not make an explicit reservation before the deadline cannot make a request afterwards. Therefore, these passengers are not explicitly considered in the optimization procedure, which only takes requests into account. However, these passengers can still go to a mandatory stop to board a bus, without the need for a reservation.

A small example of the FSMS is shown in Figure 1. In this example, a single bus is utilized for two trips, that is, bus trip A and bus trip B, to serve 13 passengers. Each passenger needs to arrive at the destination or depart from a bus stop within a certain time window. Bus trip A serves eight passengers, while bus trip B serves five passengers. There are six mandatory stops, labeled m_0 to m_5. The first mandatory stop is the start of the route of each bus trip, while the last mandatory stop is the



Legend

Mandatory stop	Bus trip A
Optional stop (visited)	Bus trip B
Optional stop (not visited)	Main route
Passenger	Cluster

Timetable

Bus stop	Bus trip A	Bus trip B
m_0	10:00	10:30
m_1	10:04	10:34
m_2	10:10	10:37
m_3	10:12	10:39
m_4	10:17	10:42
m_5	10:20	10:46

FIGURE 1 Example of the FSMS.

destination. Between two mandatory stops there is a cluster containing six optional stops. The clusters are labeled c_1 to c_5. The positions of both the mandatory and optional stops is predetermined. Whether or not an optional stop is visited depends on demand nearby. The main route, that is, the fastest route from the start to the destination that solely visits the mandatory stops, is shown as a dashed line. The bus deviates from the main route to pick up passengers. If an optional stop is visited by a bus to pick up a passenger, it is colored black. Otherwise it is shown as a white circle. Passengers are assigned to a bus trip and to a bus stop. This assignment is shown as a dotted line between a passenger and a bus stop visited by a bus on a trip. In cluster c_4 on bus trip B, it can be seen that two passengers are assigned to the same optional stop, even though that bus stop is not the closest stop to them both. This is done to reduce the travel time of the passengers onboard the bus on trip B in order to reduce the objective function further and/or to make the solution feasible. After all, a solution can become infeasible if a passenger does not arrive at the destination within their desired time window. This can occur, for example, because their bus visits too many stops and takes longer to reach the destination. After the bus on trip A reaches the destination, it is sent back to the start in order to be reused again for the subsequent trip, that is, bus trip B. Figure 1 also shows the timetable of the buses on the mandatory stops. This timetable is not fixed but depends on the demand for transportation. For example, the bus on trip A takes longer to travel from m_1 to m_2 than from m_2 to m_3 because it needs to pick up passengers in cluster c_2. By comparing the departure times of both bus trips it is clear that a bus departs at each mandatory stop in a time-interval of 30 min or less. In this example, the maximum headway or the maximum time interval between bus departures at mandatory stops is 30 min, which makes this solution feasible. Furthermore, it needs to be noted that after the bus arrives at the last mandatory stop on trip A, the bus needs 10 min to reach the first mandatory stop to start trip B. The time it takes for the bus to return to be reused for the next trip needs to be considered to obtain a feasible solution.

3.2 | Assumptions

To simplify the problem to some extent, the following assumptions are made. Passengers are assumed to be traveling alone. A group of passengers that want to travel together can make a set of identical requests, that is, the same origin location and desired arrival time or departure time. However, it is not guaranteed that all passengers will be assigned to the same bus.

The DRFS is designed for a single bus route, assuming that none of these bus stops are shared across bus lines and transfers between bus lines are not considered, as is customary for feeder services. We assume the service 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. In the case of multiple lines, a passenger located between two lines could easily be assigned first to one of the lines by a simple algorithm, and afterwards, the planning of the line can be determined. The service provider could also let the passenger choose which line is better suited. Furthermore, Vansteenwegen et al. [53] show that the majority of studies concerning feeder services solely optimize one bus line at a time. For example, Wei et al. [56] 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.

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 [18], 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 [45] 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 is possible that the service can be overcrowded in the case that there are too many passengers without a reservation. Capacity constraints for passengers with a reservation help to control the crowding of these passengers, 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 larger number of passengers without a reservation. The number of passengers without a reservation at each mandatory stop can be estimated with the use of historical data. A probability distribution for the arrival of passengers without a reservation at mandatory stops can be defined with the use of this data. Part of the capacity that is reserved for passengers without a reservation can then be defined as the mean value plus a certain number of standard deviations. This is the same principle that is used in inventory control in warehouses, where certain storage remains unused for a period of time with the expectation that inventory levels will rise in the future. An alternative would be to give priority to passengers that make a reservation and only allow passengers without a reservation if it is sure that passengers with a reservation can have a place on the bus.

Furthermore, it needs to be noted 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. It also needs to be noted that a scenario where passengers leave from the same place but have different destinations, can be optimized as well with the methods presented in this article without a larger amount of effort. However, we opted to focus on the opposite scenario for this article instead, that is, where all passengers have the same destination but different origins.

Later on, we also make the optimization problem subject to a constraint that ensures all passenger requests are accepted. It should be emphasized that the constraint of accepting all requests is introduced to make the algorithm more efficient. The alternative would be to work with a penalty, which introduces its own difficulties since the value of this penalty is difficult to determine. In practice, the service requires that requests are submitted by a specific deadline. If a certain number of requests result in insufficient capacity, service providers have time to increase the number of dispatched buses in order to still provide a feasible solution. If expanding the fleet size is not possible, the algorithm must be run without the most recent requests, which implies these requests are rejected. Requests could then be prioritized in a first come first serve manner.

3.3 | Mathematical optimization model

In this section, we discuss the mathematical optimization model of the FSMS. Table 1 gives an overview of the variables, parameters and sets that are used to describe the mathematical model. The reasoning behind the parameter and variable notation is as follows. All variables and parameters are written in *italics*. Variables are in lower case, whereas parameters are in upper case. The superscript refers to a descriptive characteristic of the parameter or variable and is written in standard text format (not italic). The indices are referred to by the subscript.

The objective z of the mathematical model is to minimize a weighted sum of different components related to the service quality. The parameters W_i are weights given to each component of the objective function and can be determined by the user. The first component (1) calculates the onboard travel time of all passengers. The second component (2) minimizes the walking time of each passenger from their origin location to their assigned departure bus stop. 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 can 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. The remaining components (3) determine the differences between the desired arrival and departure times and the actual arrival and departure times of the passengers.

TABLE 1 Notation for the MIP.

Sets	
B	Set of buses
J	Set of all possible trips of a bus
O	Set of optional bus stops
F	Set of mandatory bus stops
S	Set of all bus stops: $S = F \cup O$
P_1	Set of passengers with a desired arrival time
P_2	Set of passengers with a desired departure time
P	Set of all passengers using the service during the planning horizon: $P = P_1 \cup P_2$
Parameters	
K_c	Number of optional bus stops in cluster c
M	Number of clusters
T^{t}_{ij}	Travel time from bus stop $i \in S$ to bus stop $j \in S$
T^{w}_{pi}	Walking time of passenger $p \in P$ to departure bus stop $i \in S$
T^{arr}_{p}	Desired arrival time of passenger $p \in P_1$ at the destination bus stop $m_{\lvert F\rvert-1}$
T^{dep}_{p}	Desired departure time of passenger $p \in P_2$ at their assigned departure stop
T^{sr}	Amount of time needed to travel from m_0 to $m_{\lvert F\rvert-1}$
T^{ph}	Planning horizon for optimization
D^{f}	Maximum headway at the mandatory stops
D^{w}	Maximum value for individual walking time
D^{la}	Maximum value for arriving late
D^{ea}	Maximum value for arriving early
D^{ld}	Maximum value for departing late
D^{ed}	Maximum value for departing early
C	Capacity of the buses
W_i	Relative weight given to objective function component i
Decision variables	
x_{btij}	0-1 variables determining if bus $b \in B$, on his t^{th} trip, visits bus stop $j \in S$ immediately after visiting bus stop $i \in S$
y_{pbti}	0-1 assignment variables which assume value 1 if passenger $p \in P$ is assigned to bus $b \in B$, on his t^{th} trip, and to departure bus stop $i \in S$
a_p	Arrival time of passenger $p \in P$ at destination bus stop $m_{\lvert F\rvert-1}$
d_p	Departure time of passenger $p \in P$ at their assigned departure stop
d^{s}_{bti}	Departure time of bus $b \in B$, on its t^{th} trip, at stop $i \in S$
a^{late}_{p}	$a_p - T^{\mathrm{arr}}_{p}$ when passenger $p \in P_1$ is late
a^{early}_{p}	$T^{\mathrm{arr}}_{p} - a_p$ when passenger $p \in P_1$ is early
d^{late}_{p}	$d_p - T^{\mathrm{dep}}_{p}$ when passenger $p \in P_2$ is late
d^{early}_{p}	$T^{\mathrm{dep}}_{p} - d_p$ when passenger $p \in P_2$ is early
t^{d}_{bt}	Time at which bus $b \in B$, on its t^{th} trip, is ava ilable for departure at stop m_0
δ_{bti}	Difference between the departure time of bus $b \in B$ on its t^{th} trip at stop $i \in F$ and the departure time of the next bus

$$Min \tag{1}$$

$$z = W_1\left[\sum_{p\in P}\left(a_p - d_p\right)\right] + W_2\left[\sum_{b\in B}\sum_{t\in J}\sum_{i\in S}\sum_{p\in P} T^{\mathrm{w}}_{pi} y_{pbti}\right] \tag{2}$$

$$+ \sum_{p\in P_1}\left(W_3 a^{\mathrm{late}}_{p} + W_4 a^{\mathrm{early}}_{p}\right) + \sum_{p\in P_2}\left(W_5 d^{\mathrm{late}}_{p} + W_6 d^{\mathrm{early}}_{p}\right). \tag{3}$$

The first group of constraints deals with the routing of the buses. Constraints (4) ensure that, for the mandatory stops, exactly one arc enters or leaves. Constraints (5) ensure that, for all other bus stops, at most one arc enters or leaves any stop. 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 $m_{|F|-1}$, 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 $m_{|F|-1}$ as a predecessor.

$$\sum_{j\in S} x_{btij} = 1 \qquad \forall\, i \in F,\ b \in B,\ t \in J. \tag{4}$$

$$\sum_{j\in S} x_{btij} \leq 1 \qquad \forall\, i \in O\ b \in B,\ t \in J. \tag{5}$$

$$\sum_{l\in S} x_{btil} = \sum_{l\in S} x_{btli} \qquad \forall\, i \in S_{0,N-1},\ b \in B,\ t \in J. \tag{6}$$

$$\sum_{i\in S} x_{bti0} = 0 \qquad \forall\, b \in B,\ t \in J. \tag{7}$$

$$\sum_{i\in S} x_{btN-1i} = 0 \qquad \forall\, b \in B,\ t \in J. \tag{8}$$

A second group of constraints deals with capacities or threshold values. Constraints (9) ensure that no passenger needs to walk for a longer time than a predefined maximum value D^{w}, this is important for the passenger-stop assignment. Similarly, constraints (10) ensure that any optional stop that is farther away than a mandatory stop to a passenger, is not considered as a possible departure stop for that passenger. It needs to be noted that both sets of constraints can be dealt with as an input, that is, if $T^{\mathrm{w}}_{pi} > \min\left(D^{\mathrm{w}}, \min_{k\in F} T^{\mathrm{w}}_{pk}\right)$ then $y_{pbti} = 0, \forall\, p \in P,\ i \in S,\ b \in B$ and $t \in J$. Constraints (11) regulate the number of passengers on each bus, so that buses cannot transport more passengers than a given capacity. Constraints (12) to (13) ensure that all passengers arrive and depart within the required time window. It must be noted that parameters $D^{\mathrm{ld}}, D^{\mathrm{ed}}, D^{\mathrm{la}}$ and D^{ea} can be chosen by the service provider. Setting one of these parameters to a value of zero transforms the desired arrival/departure time of the passengers into the earliest or latest arrival/departure time.

$$T^{\mathrm{w}}_{pi} y_{pbti} \leq D^{\mathrm{w}} \qquad \forall\, p \in P,\ i \in S,\ b \in B,\ t \in J. \tag{9}$$

$$T^{\mathrm{w}}_{pi} y_{pbti} \leq \min_{k\in F} T^{\mathrm{w}}_{pk} \qquad \forall\, p \in P,\ i \in O,\ b \in B,\ t \in J. \tag{10}$$

$$\sum_{p\in P}\sum_{i\in S} y_{pbti} \leq C \qquad \forall\, b \in B,\ t \in J. \tag{11}$$

$$a^{\mathrm{late}}_p \leq D^{\mathrm{la}},\ a^{\mathrm{early}}_p \leq D^{\mathrm{ea}} \qquad \forall\, p \in P_1. \tag{12}$$

$$d^{\mathrm{late}}_p \leq D^{\mathrm{ld}},\ d^{\mathrm{early}}_p \leq D^{\mathrm{ed}} \qquad \forall\, p \in P_2. \tag{13}$$

Constraints (14) and (15) define the decision variables a^{early}_p, a^{late}_p, d^{early}_p, and d^{late}_p as positive deviations between the actual arrival or departure time and the desired arrival or departure time of the passengers. Given the objective function, in each set of constraints, one of the two variables will be zero for each passenger p in the optimal solution.

$$T^{\mathrm{arr}}_p - a_p + a^{\mathrm{late}}_p - a^{\mathrm{early}}_p = 0 \qquad \forall\, p \in P_1. \tag{14}$$

$$T^{\mathrm{dep}}_p - d_p + d^{\mathrm{late}}_p - d^{\mathrm{early}}_p = 0 \qquad \forall\, p \in P_2. \tag{15}$$

Constraints (16) and (17) define the variables d_p and a_p, the departure time and the arrival time of a passenger p, respectively.

$$\text{If } y_{pbti} = 1 \text{ then } d_p = d^{\mathrm{s}}_{bti} \qquad \forall\, i \in S,\ b \in B,\ t \in J,\ p \in P_2, \tag{16}$$

$$\text{If } \sum_{i\in S} y_{pbti} = 1 \text{ then } a_p = d^{\mathrm{s}}_{bt|F|-1} \qquad \forall\, b \in B,\ t \in J,\ p \in P_1, \tag{17}$$

Constraints (18) to (20) define the variables d^{s}_{bti} and t^{d}_{bt}. Furthermore, buses are not allowed to depart from the first mandatory stop, on any trip t, before their available departure time t^{d}_{bt}. These constraints ensure that a bus stop is not served later in time than a following bus stop in the route. This makes subtour elimination constraints unnecessary.

$$d^{\mathrm{s}}_{bt0} \geq t^{\mathrm{d}}_{bt} \qquad \forall\, b \in B,\ t \in J_0. \tag{18}$$

$$t^{\mathrm{d}}_{bt} = d^{\mathrm{s}}_{bt-1N-1} + T^{\mathrm{sr}} \qquad \forall\, b \in B,\ t \in J_0. \tag{19}$$

$$\text{If } x_{btij} = 1 \text{ then } d^{\mathrm{s}}_{btj} = d^{\mathrm{s}}_{bti} + T^{\mathrm{t}}_{ij} \qquad \forall\, i \in S_{|F|-1},\ j \in S_0,\ b \in B,\ t \in J. \tag{20}$$

Constraints (21) to (22) ensure the amount of time between two consecutive buses departing from a mandatory stop does not exceed D^{f}. These constraints model the maximum allowable headway at the mandatory stops. We denote them as the headway constraints from now on.

$$\delta_{bti} \leq d^{\mathrm{s}}_{ldi} - d^{\mathrm{s}}_{bti} \qquad \forall\, l \neq b \in B,\ d \neq t \in J,\ b \in B_{|B|},\ t \in J_{|J|},\ i \in F. \tag{21}$$

$$\delta_{bti} \leq D^{\mathrm{f}} \qquad \forall\, b \in B_{|B|},\ t \in J_{|J|},\ i \in F. \tag{22}$$

Lastly, constraints (23) ensure that every passenger is assigned to exactly one bus on a trip and one departure bus stop.

$$\sum_{b\in B}\sum_{t\in J}\sum_{i\in S} y_{pbti} = 1 \qquad \forall\, p \in P. \tag{23}$$

Constraints (24) ensure that optional stops are visited when there is at least one passenger assigned to the optional stop.

$$\text{If } \sum_{p\in P} y_{pbti} > 0 \text{ then } \sum_{l\in S} x_{bitl} = 1 \qquad \forall\, i \in O,\ b \in B,\ t \in T. \tag{24}$$

The remaining constraints determine the domains of the variables.

$$y_{pbti} \in \{0,1\} \qquad \forall\, b \in B,\ t \in J,\ b \in B,\ p \in P. \tag{25}$$

$$x_{btij} \in \{0,1\} \qquad \forall\, b \in B,\ t \in J,\ i \in S,\ j \in S. \tag{26}$$

$$d^{\mathrm{s}}_{bti},\ \delta_{bti} \in \mathfrak{R}^{+} \qquad \forall\, b \in B,\ t \in J,\ i \in S. \tag{27}$$

$$t^{\mathrm{d}}_{bt} \in \mathfrak{R}^{+} \qquad \forall\, b \in B,\ \ t \in J. \tag{28}$$

$$a_p,\ a^{\mathrm{late}}_p,\ a^{\mathrm{early}}_p \in \mathfrak{R}^{+} \qquad \forall\, p \in P_1. \tag{29}$$

$$d_p,\ d^{\mathrm{late}}_p,\ d^{\mathrm{early}}_p \in \mathfrak{R}^{+} \qquad \forall\, p \in P_2. \tag{30}$$

The number of possible trips $|J|$ can be calculated as follows. The maximum number of round trips one bus is able to perform, assuming it only visits the mandatory stops, is the planning horizon T^{ph} divided by the time needed to make such a round trip, that is, $2T^{\mathrm{sr}}$. The maximum number of trips each bus can make is then:

$$|J| = \left\lceil \frac{T^{\mathrm{ph}}}{2T^{\mathrm{sr}}} \right\rceil. \tag{31}$$

The linearized version of constraints (16), (17), (20), and (24) are given in Appendix A.

4 | SOLUTION APPROACH

In this section, solution methods to solve the optimization model of the FSMS are described. First, the model is solved using a commercial solver, namely LocalSolver. Then, a metaheuristic is presented.

4.1 | Commercial solver

In order to provide a benchmark for the results that are presented in Section 6, the optimization model is solved with a commercial solver, namely with LocalSolver. This solver makes use of local search heuristics and exact optimization techniques to solve the mathematical optimization model defined in Section 3.3. This means that the solutions that are given by this solver are not guaranteed to be optimal. However, solutions obtained by LocalSolver are expected to be of good quality. These benchmark solutions are subsequently used to assess the quality of the solutions obtained by the heuristic described in Section 4.2.

LocalSolver is chosen over an exact solver such as CPLEX or Gurobi because the heuristic local search techniques allow it to find good quality feasible solutions in a relatively short time. This is crucial for a problem as complex as the optimization problem of the FSMS. The performance of LocalSolver has been tested against Gurobi for the optimization of the capacitated VRPTW, a relatively similar optimization problem to the FSMS [10]. It is found LocalSolver consistently outperforms Gurobi, especially in instances of a larger scale. Therefore, it can be expected that the results of LocalSolver will be of good quality for the FSMS as well.

Additionally, in order to model the problem in LocalSolver, some constraints are implemented differently compared to the model presented in Section 3.3. The solver allows us to model list-variables, which are arrays in which each element appears at most once (and that can therefore have a variable length). Each route of each bus on a certain trip can be modeled as an ordered set of bus stops with an undetermined length, that is, a route is a list variable. This improves the speed of finding good feasible solutions significantly with the use of LocalSolver. Furthermore, no big-M constraints are needed since LocalSolver allows us to model "if-then-else" constraints directly. Preliminary results indicate that the implementation of the mathematical model that uses list-variables and "if-then-else" constraints in LocalSolver leads to the best solutions.

4.2 | Heuristic

In this section, the heuristic that is used to optimize the operations of the FSMS is explained. The main outline of the heuristic is explained first. This is followed by a more detailed explanation of the different components of the heuristic, namely the construction algorithm of a complete solution, the construction parameters and the local search algorithm.

The heuristic developed in this article follows what is, to the best of our knowledge, a new metaheuristic concept which we call *parameter space search* (PSS). The main idea is to use a constructive heuristic that is executed multiple times with different heuristic parameters to generate a diverse set of solutions, and combine it with a randomized local search heuristic that operates on the parameter space of the constructive heuristic. The local search heuristic attempts to discover the values for the parameters of the underlying constructive heuristic that result in the best solution quality. PSS is related to parameter tuning heuristics (like F-race, etc.), but differs from it in that it is an integral part of the optimization algorithm rather than a step to be executed before the algorithm is used.

The reason why the PSS heuristic is developed is because the optimization problem of the FSMS is quite complex. We also consider this type of heuristic as a methodological contribution that may serve to inspire others. The optimization of the FSMS involves different interdependent decisions. This problem can be viewed as an integration of a VRP, an assignment problem, and a highly constrained timetabling problem. All passengers are assigned to a bus and to a departure bus stop, taking into account the maximum walking time and their preferred departure or arrival 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, a bus must arrive within a time interval at each mandatory stop and buses need time to return to the first mandatory stop in order to be reused for a next trip. All these decisions are intertwined and affect one another. In particular, bus trips cannot be determined independently from each other since the headway constraints can only be respected by determining the timetable of all bus trips. All these complexities make it very difficult to find feasible solutions. By iteratively constructing new solutions from scratch, the PSS heuristic overcomes the complexity of the interdependency of bus trips. The semi-randomness of the PSS heuristic offers a good way to bring variability in the construction of bus trips and increases the chance of finding more feasible solutions and higher quality solutions.

4.2.1 | Main outline of the heuristic

The outline of the PSS heuristic is shown in Algorithm 1. The heuristic constructs a complete solution in each of its iterations i, which we denote as construction iterations. A complete solution serves all passengers and consists of a set of buses that make a number of trips. A bus trip corresponds to the journey of a bus from the starting point to the destination. Since buses return to the starting point to be reused, a single bus can have several trips. The construction of a complete solution in each construction iteration i is denoted as `Construct_Solution` in Algorithm 1. `Construct_Solution` contains, among others, the generation of many bus trips. The generation of a single bus trip consists of assigning passengers to a bus b on a certain trip t and constructing its route and timetable. The generation of a bus trip is denoted as a trip generation. This trip generation is explained in more detail in Section 4.2.2. When no more bus trips can be scheduled within a certain planning horizon, the solution is complete and `Construct_Solution` ends.

The construction algorithm `Construct_Solution`, in iteration i, is greedy because it aims to construct the best routes, timetables and assignments for one bus trip at a time regardless of the next bus trips. This means that there are instances where the resulting solution is infeasible because not all passengers are assigned to a bus. Due to the tight constraints of the optimization model, infeasible solutions can often not be restored to feasibility without constructing the solution from scratch again. For this reason, these infeasible solutions are discarded. The PSS heuristic executes N^{stop} construction iterations, after which the best solution, that is, the solution with the lowest objective function value, is kept. If no feasible solutions are found after N^{stop} construction iterations, the instance itself is considered to be infeasible. An assumption of the FSMS is that all passengers must be served. This might, of course, lead to infeasible solutions if the capacity of the system is insufficient to serve all passengers. In such cases, the decision maker can relax this constraint, and, for example, attempt to find the solution with the most passengers served. Such alternative formulations of the FSMS, with different constraints and objective function are, however, left for future research.

To bring more variance into the construction of the solutions, construction parameters are introduced. These parameters guide `Construct_Solution` and bring a balance between the greediness and the feasibility of the construction algorithm by determining whether or not a certain passenger is assigned to a bus trip. This allows the algorithm to find more solutions with a better objective value or to find more feasible solutions for "difficult instances," that is, instances where it is difficult to find feasible solutions due to strict constraints. The mechanism of these construction parameters is further explained in Section 4.2.3.

The values of the construction parameters are randomly sampled until values that lead to a feasible solution are found. Afterwards, these values are changed for each construction iteration i, which leads to different solutions in each construction iteration i. The goal of the PSS heuristic is then to find the best values of the construction parameters. This is done with local search, and more specifically with SA. As a result, the different construction iterations are not independent, since valuable information about the construction of a good solution is used to construct the next solution. The local search is further explained in Section 4.2.4. The local search is also used to automatically fine-tune the parameters of the underlying greedy construction algorithm.

Algorithm 1. Main outline of the PSS heuristic

```
i = 1
while i ≤ N^stop do
    if no feasible solution found then
        Search randomly for feasible construction parameters r^c1, ..., r^cm;        // See Section 4.2.3
        Incumbent solution = Construct_Solution(r^c1, ..., r^cm) ;                   // See Section 4.2.2
        Incumbent construction parameters r^b1, ..., r^bm ← r^c1, ..., r^cm
    else
        Sample construction parameters r^c1, ..., r^cm in the neighborhood of the incumbent construction parameters
         r^b1, ..., r^bm ;                                                           // See Section 4.2.4
        Candidate solution = Construct_Solution(r^c1, ..., r^cm) ;                   // See Section 4.2.2
        if new solution is feasible then
            ΔE = objective function value candidate solution - objective function value incumbent solution
            if ΔE < 0 then
                Incumbent solution ← candidate solution
                r^b1, ..., r^bm ← r^c1, ..., r^cm ;                                  // See Section 4.2.4
            else if exp(−ΔE/T) ≥ Uniform(0,1) then
                Incumbent solution ← candidate solution
                r^b1, ..., r^bm ← r^c1, ..., r^cm ;                                  // See Section 4.2.4
            end
            Update temperature T according to a cooling schedule ;                   // See Section 4.2.4
            i++
        end
    end
end
Return incumbent solution
```

4.2.2 | Construction algorithm of one construction iteration

The pseudo-code for the construction algorithm `Construct_Solution` is given in Algorithm 2 in Appendix B, together with the pseudo-code of the associated functions (Algorithms 3–6).

Preprocessing

Before the start of the construction algorithm, the route *BR* is determined. This is a route that is constructed with the aim of minimizing the total travel time of a bus that visits all bus stops. This route is constructed in order to insert additional optional stops in the best possible position in existing routes later on, when bus trips are generated. First, an initial feasible route is constructed and afterwards it is improved with a 2-opt procedure. The initial route is constructed by adding all the mandatory stops first and then arbitrarily adding optional stops between the mandatory stops. The 2-opt algorithm implemented here is a first-improvement algorithm, and selects two edges of the existing route and swaps them if and only if the objective function value is lowered by this swap. This procedure is done once before the start of the first construction iteration.

Initialization

In the first step of a construction algorithm, the passengers with a desired arrival time T_p^{arr} are placed in a queue Q^{a} and are sorted according to their T_p^{arr}. Passengers with a desired departure time T_p^{dep} are placed in a queue Q^{d} and are sorted as well. Later on, passengers are considered for a bus assignment in the order of queues Q^{a} or Q^{d}. The queues are sorted according to the desired arrival or departure times of the passengers because passengers with comparable desired arrival and departure times are more likely to benefit from being assigned to the same bus. The first available departure times t_b^{d} of each bus b are set to the start of the planning horizon in order to have initial feasible departure times.

Loop of trip generations

In the second step, the construction algorithm enters a loop, in which bus trips are generated. We denote each iteration of this loop as a trip generation. In each trip generation, the earliest available bus b, that is, the bus with the lowest t_b^{d}, is considered.

The variable t_b^d is dynamically updated in each trip generation. This means a different bus is considered in each trip generation. The number of trip generations is not known beforehand and depends on the parameters of the instance, such as the number of passengers or the fleet size.

In each trip generation u, that considers bus b on trip t, an initial route for bus b is created, in which all the mandatory stops are visited. An initial timetable is constructed as well, in which the bus leaves as soon as possible, that is, at time t_b^d. This route and timetable are subsequently updated as a passenger p is assigned to bus b on trip t. For each trip generation u, the passengers in either Q^a or Q^d are considered. To determine which queue is considered, the earliest desired arrival time T^a of Q^a and the earliest expected arrival time T^d of Q^d are determined. The latter is calculated as the smallest T_p^{dep} plus the time it would take to go from m_0 to $m_{|F|-1}$ by only visiting the mandatory stops. If $T^a \leq T^d$ and Q^a is not empty, the passengers from Q^a are considered first. Otherwise, if $T^a > T^d$ and Q^d is not empty, the passengers from Q^d are considered first. This is done in order to address the most urgent requests first. In both cases, passengers are iteratively added to the bus trip as long as capacity constraints are still feasible and while the queue is not empty. After the timetable and the route of bus b on trip t is determined, the timetable can be modified to accommodate additional passengers from both queue Q^a and queue Q^d. This is done with three construction parameters and is explained in more detail in Section 4.2.3. When no more passengers from a queue can be assigned to bus b on trip t, the trip generation u ends and the next bus is considered.

In case Q^a is considered, the function `Best_Stop_a` is used to determine the best possible departure stop s for the current passenger p from queue Q^a. This is done in a manner similar to the pilot method metaheuristic framework [54], where better solutions are found by looking ahead to see which possible choice worsens the objective the least. In our heuristic, the best departure stop s is determined by calculating the additional cost of each feasible departure stop and choosing the departure stop with the lowest cost. The additional cost is the sum of two factors: the additional travel time that the bus needs to travel to reach the bus stop, and the walking time of the passenger walking to the bus stop. A bus stop is feasible if it is within D^w walking time. Furthermore, if adding a departure stop to the route makes the headway constraints or the desired arrival time window constraints infeasible, the bus stop is not feasible either. If there are no feasible bus stops available, the passenger is not assigned to the bus. If there exists a feasible departure stop, then the best feasible departure stop is assigned to passenger p on bus b on trip t.

Afterwards, if the departure stop is not part of the route already, the route of the bus is updated with function `Update_route`. This is done with the help of the route BR, the highly efficient route that visits all bus stops, determined during preprocessing. The departure bus stop s is inserted in the current route depending on where s is placed in BR. For example, if BR is $[m_0, o_0, o_1, m_1, o_2, o_3, m_2]$, the current route is $[m_0, m_1, m_2]$ and we wish to insert bus stop o_2 in the current route, then it should be inserted between m_1 and m_2 because that is the most similar placement of o_2 in BR. By updating the route in this manner, we can assure that the route remains efficient, without the need to perform heavy time consuming improvements on it. Since part of the route is fixed, this method of updating the route yields good results.

After the route is updated, the timetable is updated with `Update_timetable`. Since the difference in the desired arrival time T^{arr} and the actual arrival time, that is, the departure time $d_{|F|-1}$ at mandatory stop $m_{|F|-1}$, is a sum of absolute deviations, 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 $[LB_b, UB_b]$, which is dependent on the desired arrival times. 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 $\sum_p |T_p^{arr} - d_{|F|-1}|$ is a convex function of $d_{|F|-1}$. If the resulting departure times of the bus at the mandatory stops make the headway constraints infeasible, the arrival time can be adjusted in order to satisfy the constraints. If it is still not possible to satisfy both the passenger time window constraints and the headway constraints, the passenger is not assigned to the bus. If the arrival time adjustment results in the bus leaving the first mandatory stop before the bus is available for departure, the arrival time can be corrected as well. In case this is in conflict with the other constraints, the passenger is not assigned to the bus either. Whether or not these last two timetable adjustments take place depends on the construction parameters, this is explained in more detail in Section 4.2.3.

In case Q^d is considered, the function `Best_Stop_d` is used to determine the best possible departure stop for the current passenger. This function determines the best stop in the same way as `Best_Stop_a`, however, the difference in desired and actual departure time of the passenger is also added to the additional cost of the potential passenger-bus stop assignment. This is done because for passengers in Q^d, the departure stop affects the difference in departure times. Moreover, there are additional restrictions present. Bus stops cannot be visited before the previously inserted bus stop in the current route. This is because the passengers are added to the bus according to an ascending T_p^{dep}, which means that each subsequent passenger wishes to depart at a later time. For example, assume that passenger p_1 wishes to depart at a later time than passenger p_0, but p_1 is assigned to a bus stop that is visited before the departure stop of p_0 in the current route. The bus is then forced to pick up passenger p_0 before passenger p_1 due to the departure time window constraints. However, the departure bus stop of p_1 is visited before the departure time of p_0, which makes this passenger-bus assignment inefficient due to backtracking or even infeasible in most cases. In some cases, such an assignment might be possible. For example, if the departure bus stops of both passengers are close to each other

TABLE 2 Overview of construction parameters.

Parameter	Range	Determines	When?
r_u^{d}	[0.5, 1.0]	Maximum headway for bus trip generation u	During the generation of a bus trip
r_u^{f}	[0.0, 1.0]	Maximum timetable adjustment for feasibility	
r_u^{w}	[0.0, 1.0]	Maximum walking time for additional passengers	After the generation of a bus trip
r_u^{sf}	[0.0, 1.0]	Maximum forward timetable shift	
r_u^{sp}	[0.0, 1.0]	Maximum backward timetable shift	

and the maximum allowable time to depart earlier than the desired departure time is large enough for the bus to drive from one stop to another. However, this is not a likely scenario and such an assignment is quite inefficient, thus it is not considered.

If the bus cannot possibly arrive at a bus stop within the desired departure time window of the passenger, the bus stop is infeasible for this passenger-bus assignment. The headway constraints and the bus departure constraints from the first mandatory stop are taken into account in the same manner as in the function `Best_Stop_a`. If there are no feasible bus stops available, the passenger is not assigned to the bus. If there are feasible bus stops, the route and the timetable are updated with `Update_route` and `Update_timetable`. The function `Update_timetable` determines the median of the T_p^{dep} of the passengers onboard.

The loop with all trip generations stops only if the departure time of the current bus at the last mandatory stop is outside of the planning horizon or if all passengers have been assigned to a bus trip.

Initialization for the next construction

After all possible passengers are added to the bus, the bus becomes available for the next trip and the earliest available time t_b^{d} of the bus is updated. The new t_b^{d} is calculated as the arrival time at $m_{|F|-1}$ plus the time T^{r} that the bus needs to reach the first mandatory stop again. If no passengers are assigned to a bus, the bus will be empty and it will visit all mandatory stops in order to fulfill the headway constraints.

4.2.3 | Construction parameters

The construction algorithm in Section 4.2.2 is greedy, which can lead to infeasible solutions. The construction algorithm can be too greedy by, for example, optimizing the routes and timetables of the first 20 passengers without taking into account the remaining requests. As a result, passengers that are already processed in the algorithm are provided with a good service. However, the remaining requests cannot be processed anymore, for example, because there are no more buses available at the right time. Therefore there is a need to balance the search for a feasible solution and the greediness of the construction. With this goal in mind, several sets of construction parameters are introduced.

An overview of the construction parameters is shown in Table 2. This table shows what each construction parameter determines and when in the algorithm this takes place. The table also shows the range of values each parameter can take. Two of these parameters are used during the generation of a bus trip, while the others are used after each bus trip generation. These parameters are further explained in more detail.

The construction parameters have an index u, which refers to a trip generation. This is the trip generation where the current bus b is dispatched on trip t and for which the construction algorithm is determining the planning. For example, $u = 1$ can refer to bus $b = 0$ on trip $t = 1$ and $u = 2$ refers to the next bus that is dispatched, for example, bus $b = 5$ on trip $t = 0$. Each set of construction parameters has N^{rd} number of elements. This number corresponds with the number of bus trips that are needed in each construction iteration and depends on how the solution is constructed. In each construction of a complete solution each set is unique. To illustrate this, the values of r^{f} for two different construction iterations i and $i + 1$ are shown in Figure 2. Each bar in the figures corresponds to the construction parameter r_u^{f}, of the set r^{f}, on trip generation u. It can be seen that in construction iteration i, the set r^{f} has $N^{\mathrm{rd}} = 20$ elements while in construction iteration $i + 1$ it has 21 elements. Furthermore, the values of each element in each set fluctuate from bus trip to bus trip. These parameters can be viewed as variables of a function that constructs a complete solution and the variables that lead to the best solution need to be determined.

The construction parameter r_u^{d} is used to determine the maximum allowable time D_{bt}^{f} between the departure of bus b on trip t and the previous bus at any mandatory stop. This is simply done by multiplying the maximum allowable time D^{f} with r_u^{d}, the resulting product is then D_{bt}^{f}. The parameter r_u^{d} can take values between 0.5 and 1.0. This means that $D_{bt}^{\mathrm{f}} \leq D^{\mathrm{f}}$ for all buses and trips, and D_{bt}^{f} is unique for each bus trip. Limiting the headway constraint further to a different extent for each bus separately allows the construction algorithm to increase the number of different solutions it can create by bringing more variability to the construction.

With each passenger-bus assignment there is a choice to adjust the timetable to either make the headway constraints feasible or to make the departure times feasible or both. Whether or not these adjustments take place depends on the construction

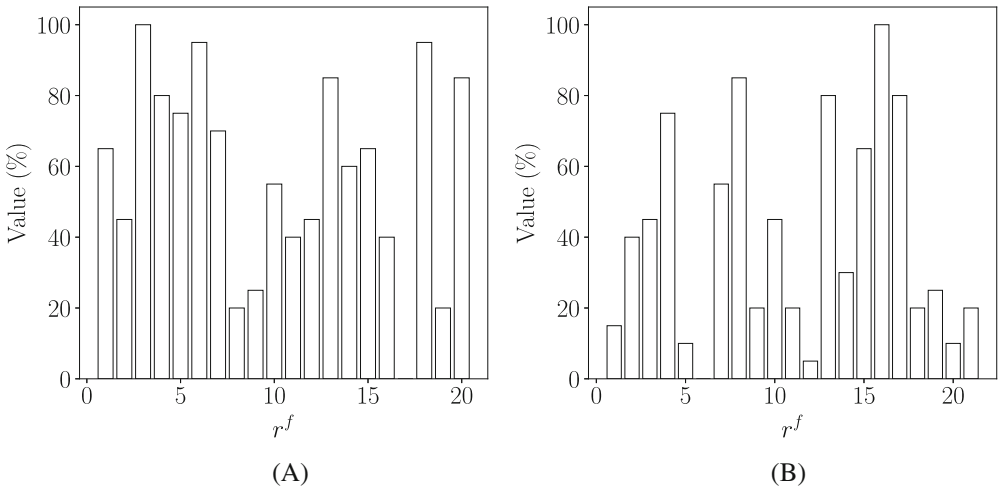


FIGURE 2 Values of the set r^{f} for two different construction iterations. (A) Values for r^{f} on construction iteration i. (B) Values for r^{f} on construction iteration $i + 1$.

parameter r_u^{f}. The construction parameter r_u^{f} can take any value between 0 and 1. The feasibility ratio R^{f} is the percentage between the ideal cost increase of adding passenger p to bus b on trip t, for example, the cost increase if there were no headway constraints or passenger departure time window constraints, and the current feasible cost increase. If r_u^{f} is smaller than feasibility ratio R^{f}, the passenger is not assigned to the bus. In case the passenger is not accepted for this assignment, the passenger is still considered for an assignment in the next bus trip generation. If there are no more bus trips available, the passenger is accepted regardless of the feasibility ratio. This is a way to balance the feasibility and the greediness of the construction. Since r_u^{f} is different in each construction iteration, this can lead to different solutions in each construction of a complete solution.

During the trip generation both the route and timetable are gradually constructed from scratch. After a trip generation, only the timetable can be modified to accommodate additional passengers from both queue Q^{a} and queue Q^{d}. This is done with the last three construction parameters r_u^{sp}, r_u^{sf}, and r_u^{w}. To accommodate more passengers, the maximum amount of time D^{sp} that the buses can depart earlier in time than originally planned, or the maximum amount D^{sf} that buses can depart later in time, is determined. The desired arrival or departure times of the passengers onboard and the headway constraints together create feasible time windows for the departure of the bus at each bus stop. We can then create two lists L^{l} and L^{u}, containing the differences in time between the current departure time of bus b and respectively the lower and upper bound of the feasible time window of each visited stop. The parameters D^{sp} and D^{sf} are then the minimum values of respectively L^{l} and L^{u}. Adding an additional passenger p, makes the timetable shift T_p^{s} amount of time in order to make the assignment feasible. Here, T_p^{s} can be negative or positive. A passenger p is assigned to the bus if the following conditions are satisfied. First, if T_p^{s} is negative, the absolute value needs to be lower than $r_u^{\mathrm{sp}} D^{\mathrm{sp}}$. Second, if T_p^{s} is positive, then it needs to be lower than $r_u^{\mathrm{sf}} D^{\mathrm{sf}}$. Lastly, the passenger cannot walk more than $r_u^{\mathrm{w}} D^{\mathrm{w}}$ amount of time to their assigned bus stop and their assigned bus stop must already be part of the route. The parameters r_u^{sp}, r_u^{sf}, and r_u^{w} are construction parameters and can take values between 0 and 1. These parameters limit the number of additional passengers that are added to a bus, that is, passengers added after the trip is generated. This increases the greediness of the construction by only accepting assignments with a certain ideal level of quality. Reducing the maximum walking time, for example, leads to more greedy solutions since we only accept the more ideal assignments that lead to less walking time. This provides more variability in the construction of a complete solution, which leads to better solutions and a larger number of feasible solutions.

4.2.4 | Local Search for construction parameters

In the previous section, various construction parameters are used in order to guide the construction of a single complete solution. In this section, the local search algorithm that is implemented to find the best values for these parameters is discussed. For the remainder of this section, we refer to the construction parameters r^{d}, r^{f}, r^{sp}, r^{sf}, and r^{w} as variables and a set of values for these variables as a solution.

The local search framework that is used in this article is the framework of SA. The main idea behind SA is allowing occasional uphill moves to avoid entrapment in poor local optima. The search starts from an initial feasible solution, that is, an initial value for each construction parameter. Each solution has a specific objective function value. A small change in one or several variables can generate a neighboring solution with a different objective value. The neighboring solution is generated by randomly sampling different variables in the neighborhood of the incumbent solution. If the cost value of the candidate solution is better than that of the incumbent solution, a move to the candidate solution is made. However, if the candidate does

not improve the incumbent solution, there is still a chance of transition. The probability of accepting such an uphill moves is modeled with the Boltzmann distribution $\exp\left(\frac{-\Delta E}{T}\right)$, with ΔE the difference in objective function value between the candidate and the incumbent solution and T the current temperature. In practice, if a randomly chosen number between zero and one is smaller than this Boltzmann factor, an uphill move is accepted. The temperature typically is given a high value at the start of the search, in order to move out of local minima. Afterwards, the temperature is decreased according to a cooling schedule, which decreases the probability of accepting an uphill move as the algorithm continues. In the literature, there are several types of cooling schedules, categorized into classes such as monotonic schedules, adaptive schedules, geometric schedules, and quadratic cooling schedules. In this article, we consider the cooling schedule presented by Azizi and Zolfaghari [4]. A traditional cooling schedule, such as the geometric decrease of the temperature, is useful if the local minima are near the start point. However, this may not lead to a near optimal solution if some local minima are encountered at a relatively low temperature towards the end of the search. Azizi and Zolfaghari [4] propose an adaptive SA method that takes into consideration the characteristics of the search trajectory. In this method, the temperature can be increased and decreased depending on the trajectory of the search. The following temperature control function is used:

$$T_i = T^{\min} + \lambda \ln(1 + \delta_i). \tag{32}$$

Here, λ is a coefficient that controls the rate of temperature rise, $T^{\min}$ is the minimum value that the temperature can take and δ_i is the number of consecutive upward moves at construction iteration i. The initial value of δ_i is zero, thus the initial temperature $T_0 = T^{\min}$. More concretely, δ_i is given by:

$$\delta_i = \begin{cases} \delta_{i-1} + 1 & \Delta E > 0 \\ \delta_{i-1} & \Delta E = 0 \\ 0 & \Delta E < 0. \end{cases} \tag{33}$$

The rationale behind this approach is that downhill moves are more common in the beginning of the search, which means that there is less need for a high temperature to push the search out of local minima. However, as the search continues, the chance of getting trapped in a local minimum increases. Therefore, a high temperature that could move the search out of local minima is needed. Another benefit of this method is that the search continues for a predetermined number of construction iterations N^{stop} and does not freeze if the computational time is extended. In our case, N^{stop} refers to the number of feasible construction iterations.

The variables that are used to construct the first feasible solution are randomly sampled until a feasible solution is found. The consecutive candidate solutions are sampled in the neighborhood of the incumbent solution. Each element of each set of the candidate variables is sampled with a normal distribution with a mean value μ equal to the current variable value, and with standard deviation σ. The latter is a parameter of the SA local search and determines the size of the neighborhood we look for candidate solutions. Furthermore, there are many possible solution compositions since there are many variables that constitute a solution. The local search may require a large number of construction iterations to provide good quality solutions because a large portion of the solution space needs to be explored. In order to limit the generation of infeasible solutions, we do not re-sample all variables in each construction iteration of the PSS heuristic. Instead, a subset of the variables remains the same, that is, the values of these variables remain the same as in the previous construction iteration. For example, the sampling of a candidate variable $r_u^{c,\text{f}}$, that is, the new value of r_u^{f}, in the next construction iteration, in trip generation u, is then given by:

$$r_u^{c,\text{f}} = \begin{cases} \text{Normal}\left(r_u^{b,\text{f}}, \sigma\right) & r_u^{c,\text{f}} \in R^{\text{c}} \\ r_u^{b,\text{f}} & \text{otherwise.} \end{cases} \tag{34}$$

With R^{c} the set of variables randomly chosen to be changed and $r_u^{b,\text{f}}$ the incumbent variable of r_u^{f}. It needs to be noted that sampling the variables with a normal distribution can lead to instances where the variables take invalid values. In such cases, the value of the variable is reset to the closest valid value. The number of variables $|R^{\text{c}}|$ that are changed in each construction iteration, the neighborhood size σ, the coefficient λ and the minimum temperature $T^{\min}$, are parameters of the SA and need to be fine tuned. This is done in the next section.

5 | EXPERIMENTAL SET-UP

In this section, the instances that will be used to evaluate the performance of the PSS heuristic are discussed. Afterwards, the best parameters for the local search are determined. The heuristic is run on a computer with a Windows 10 Enterprise operating system, an Intel Core™ i7-8850H, 2.60GHz CPU and 16 GB of RAM.

5.1 | Instances

To test the PSS heuristic, different instances are used. Instances I_1 to I_{15} are benchmark instances that are solved with LocalSolver as well. The positions of the mandatory stops are chosen arbitrarily and are equidistant. The optional stops are scattered around a location between the mandatory stops. The positions of the passengers are randomly chosen within a certain radius of the bus stops. The passenger desired arrival or departure times are randomly sampled in alternating time-intervals within a planning horizon of four hours. All instances are listed in Table C1 in Appendix C. The instances have different attributes, namely the number of buses $|B|$, the number of requests $|P|$, the number of bus stops $|S|$, the maximum headway at the mandatory stops D^{f} (in minutes) and the vehicle capacity C. There is one cluster between two mandatory stops, so there are $M = |F| - 1$ clusters. Furthermore, without loss of generality, each cluster has the same number of bus stops K. The first half of the $|P|$ passenger requests have a desired arrival times and the other half have a desired departure times. All the objective weights W_i are given equal value, in order to give each component of the objective function equal importance. A detailed study of the effect of the weight values on the solutions is beyond the scope of this article. The details of the parameters of the instances, as well as the solutions discussed in this article are available in detail online: https://www.mech.kuleuven.be/en/cib/drbp/mainpage#section-8.

5.2 | Fine-tuning local search parameters

The parameters of the local search need to be fine-tuned in order to ensure that the heuristic gives good quality results consistently within a short amount of time. For the remainder of this section, a representative subset TS^{a} of 15 of these instances is selected. These instances are indicated in bold in Table C1.

For each of the instances in TS^{a}, the heuristic is run with different values of each local search parameter. To reiterate, the local search parameters are: the number of variables $|R^{\mathrm{c}}|$ that are changed in each construction iteration, the neighborhood size σ, the coefficient λ and the minimum temperature $T^{\min}$. Different local search parameter values result in different objective function values and different runtimes. Only one parameter is changed at a time, in order to isolate their effect on the results. However, even with the same set of parameter values, different objective values may result from the heuristic, due to the randomness of the algorithm. To obtain an adequate estimate of the objective function value, the heuristic is run 30 times for each set of local search parameter values for each instance. The algorithm runs for $N^{\mathrm{stop}} = 30\,000$ feasible construction iterations on each run.

Both the resulting mean value of the objective values and the runtimes are considered in the choice of these parameters. Generally, it is found that values that increase the runtime also improve the quality of the solutions. Choosing a larger neighborhood coefficient σ leads to better solutions on one hand because a larger part of the solution space is explored. On the other hand, choosing a larger neighborhood results in longer runtimes because there are more infeasible solutions found and discarded. Choosing to re-sample a larger number $|R^{\mathrm{c}}|$ of variables leads to better solutions and a longer runtime as well. Similarly, when more variables are re-sampled, there is a higher chance to encounter infeasible solutions, with the trade-off of exploring more of the solution space. Choosing smaller values for λ or $T^{\min}$ leads to better results and slightly higher runtimes in some instances. Lower values for these parameters result in a slower temperature change, which helps to escape local minima. The runtime likely increases in some instances due to the fact that at lower temperatures, the algorithm is near local minima and no uphill moves are allowed often. These instances may have a small number of feasible solutions near the local optima, which prolongs the search. In some cases, the decrease in objective value is not significant compared to the increase in runtime. For example, Figure 3 shows the influence of the λ and σ parameters for instance I_{17}. It is clear that the objective does not improve by much for $\sigma > 0.15$ or for $\lambda < 10$. This is thus a good criteria to determine the best parameter values.

From these experiments and with these criteria in mind, it is concluded to set $\sigma = 0.125$, $\lambda = 10$, $|R^{\mathrm{c}}| = 10$ and $T^{\min} = 0.01$. These are the values that gave the best range of objective values in the shortest times for all of the instances from TS^{a}.

6 | PERFORMANCE OF THE ALGORITHM

In this section, the results for all the instances that are discussed in Section 5 are presented. In the previous section, the parameters of the SA algorithm were determined on a training set of instances. First, the performance of the heuristic on a small set of instances is discussed. These instance are run several times in order to gain insight on the behavior of the heuristic. Afterwards, we compare the results of using three different local search methodologies to find the best construction parameters. Finally, all instances are evaluated.

6.1 | Behavior of the heuristic

Figure 4 shows the (A) accepted and the (B) improving feasible solutions that are found during an individual run on instance I_{19}. Each dot in this figure represents an objective function value that corresponds with a feasible solution found by the heuristic during the construction iterations.

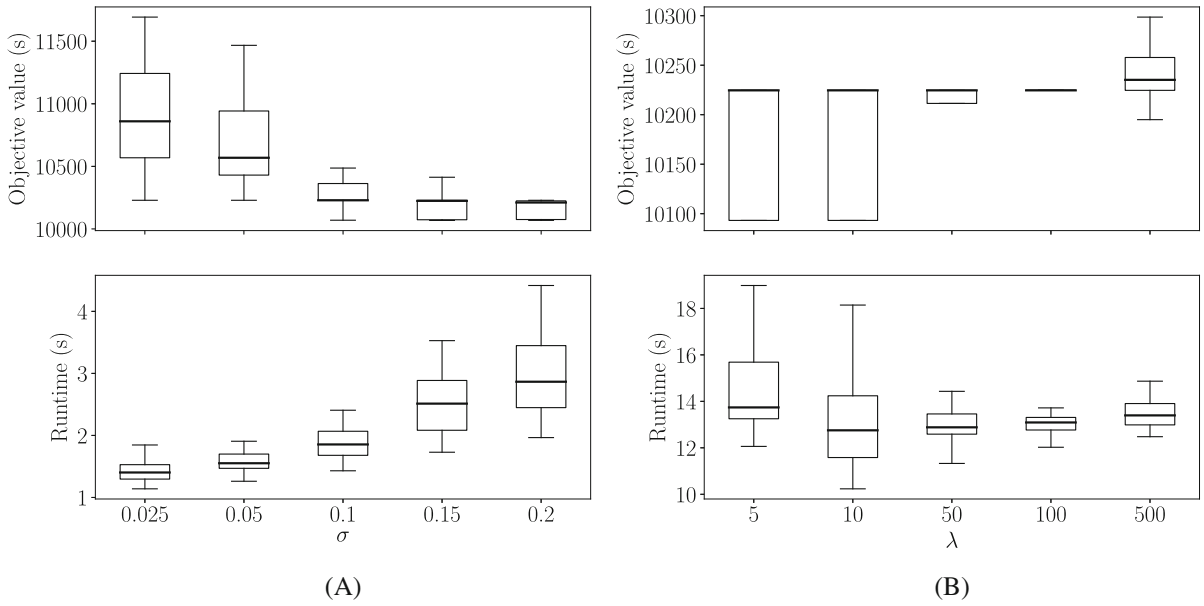


FIGURE 3 Influence of local search parameters (B) λ and (A) σ on instance I_{17}.

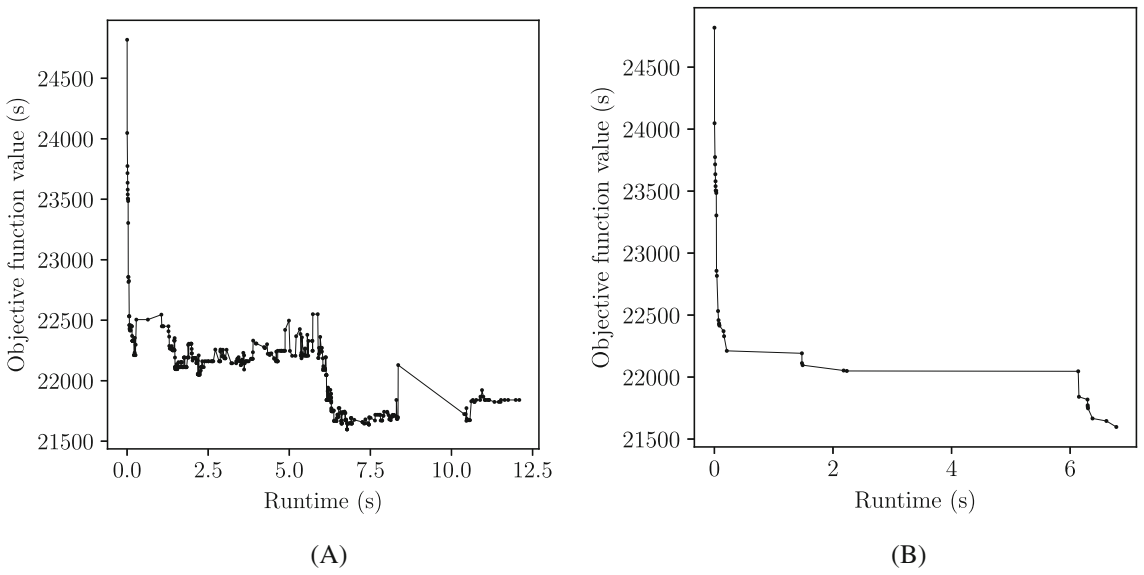


FIGURE 4 Feasible solutions found by the heuristic. (A) Accepted solutions. (B) Improving solutions.

The heuristic makes most of its improvements at the start, within the first seconds. Within these first seconds of runtime, the first feasible solution is found. It can be seen that afterwards the heuristic accepts solutions with a higher objective function value in order to escape possible local minima. After a certain point in time, approximately after two seconds, the heuristic is not able to find better solutions so easily. This is likely due to the presence of an isolated local minima, that is, a local minima that is difficult to escape from. However, with the SA procedure the heuristic is able to escape this local minima to reach a lower objective function value. After approximately 8 s, the heuristic seems to not find any feasible solution at all for some time. It is possible that the feasible solution space is quite narrow around the incumbent solution. This means that infeasible solutions are found more often and more construction iterations with different parameters are needed to find something feasible again. The total runtime of the heuristic on instance I_{19} is 12.4 s, while the last improvement is found after approximately 7 s. This means that a significant portion of the runtime is devoted to reaching the stopping criterion: 30 000 construction iterations that yield feasible solutions. In other words, the algorithm can sometimes yield good results even when the required number of construction iterations is lowered.

It needs to be noted that the time it takes to find the first feasible solution is dependent on the instance. In general, in most instances, the heuristic finds a feasible solution within the first second of runtime. Some instances which are more difficult to solve, such as instances I_{18}, I_{33}, I_{36}, and I_{40}, take a relatively longer time to find the first feasible instance (up to a few seconds).

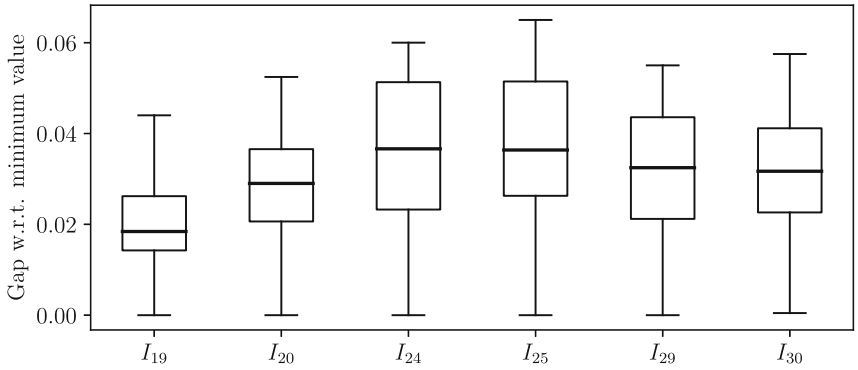


FIGURE 5 Box-plot of obtained solutions after 100 heuristic runs.

These instances are characterized by having more stringent constraints, such as a low number of buses compared to the number of requests, low vehicle capacities and so forth. Furthermore, the same instance may take longer to find feasible solutions on one run compared to another. The time it takes to find feasible solutions is also determined by the stream of random numbers.

Instances $I_{19}, I_{20}, I_{24}, I_{25}, I_{29}$, and I_{30} are run 100 times to observe the different objective function values that the heuristic produces when different streams of random numbers are utilized. Figure 5 shows a box-plot of the results that are obtained from the 100 runs. The gap, with respect to the lowest value that is observed, is reported rather than the absolute objective function value in order to compare the results of the different instances directly. Although the lowest values are not obtained often, the algorithm still guarantees solutions of good quality. For the smaller instances, solutions with gaps of 1.5% to 3.5% are observed the most. For the largest instances this goes up to the 3.5% to 5% range. Smaller instances, that is, instances with fewer decision variables seem to perform relatively better. This is expected since a smaller portion of the solution space of these instances can be explored with the same number of feasible construction iterations. Increasing the number of feasible constructions iterations for the stopping criterion can improve the overall quality of the solutions, albeit at the cost of longer runtimes. The objective function values are also quite consistent, with a relative standard deviation of 0.02 or less for all instances. Furthermore, finding lower objective function values generally does not require a longer runtime. The runtime seems to be independent from the objective function value. This means that a single better quality solution is obtained when the stream of random numbers is favorable and not necessarily when more feasible construction cycles are performed. More construction cycles, however, may improve the consistency of finding good quality solutions.

6.2 | Local search methodologies

Construction parameters guide the construction of a solution. Different configurations of these parameters lead to different, and often better, solutions. Therefore, as previously stated, finding the best construction parameters is essential. The best values for these parameters are found with the use of local search, namely with SA. The question arises whether or not the use of local search techniques is needed to obtain better quality results. We conduct the following experiments to justify the use of SA. We choose instance I_{18} for these experiments since it is a mid-size instance that is more difficult to solve, that is, an instance with more stringent constraints. The instance is optimized with the SA local search, a steepest descent (SD) method and with a random search method. The SD method works similarly to the SA method, with the difference being that in SD only downhill moves are allowed, that is, only solutions with a lower objective function value are accepted. In the random method, the construction parameters are randomly sampled in each construction iteration. Instance I_{18} is optimized with all three methods for 50 runs. All methods have a stopping criterion of 30 000 construction iterations. Figure 6 shows the results as box-plots. It can be seen that SA is the best performing methodology, followed by SD and then the random method. Both the median values and the minimum values of SA are lower than all values in both SD and the random method. Furthermore, the range of values obtained by the SA method is smaller than both the SD method and the random method, which indicates that there is less variability in the results. This indicates that SA is an appropriate method for the search of construction parameters. The reason why SA outperforms SD is likely because SA also allows uphill moves, which make it possible to escape local minima. Both SD and SA outperform the random methods because, unlike the random search, both of these methods utilize information of the previous construction iteration to find better quality solutions.

6.3 | Experimental results

The heuristic is used to solve all of the instances that are described in Section 5. The results are shown in detail in Table D1 in Appendix D and summarized in Table 3. For the first 15 instances, the problem is solved with LocalSolver as well in order to

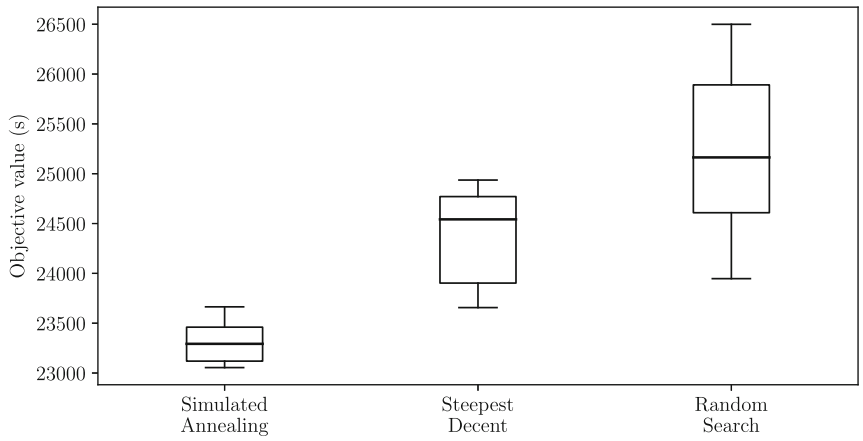


FIGURE 6 Box-plot of solutions of I_{18} obtained by three different local search methodologies.

TABLE 3 Summary of computational results.

Instances	Gap	Runtime LocalSolver	Runtime serial	Runtime parallel
I_1–I_{15}	[−4%, −21%]	3600	[1.76s, 7.14s]	[0.75s, 5.82s]
I_{16}–I_{42}			[3.81s, 323s]	[2.45s, 126s]

assess the quality of the solutions. These instances are run for one hour with the commercial optimization solver. This allows us to determine the gap between the values obtained by our heuristic and the values obtained by LocalSolver. The first row of Table 3 shows the range of values for the first 15 benchmark instances. The second row shows the range of values for the remaining larger instances.

The mean runtime of 10 runs is given in Table D1 as well, Table 3 shows a range of runtimes. The framework of the heuristic allows us to implement it in parallel as well. To do this, each semi-random greedy construction iteration is executed on a different thread, until a feasible solution is obtained. Once a feasible solution is found, the candidate and the incumbent solution are evaluated and the heuristic moves on to the next construction iteration of the local search. This means that the algorithm is sped up because feasible solutions are found in a shorter time. The runtimes of the serial and the parallel implementation with 12 threads of the heuristic are given in Table 3.

From the detailed results, it can be concluded that the heuristic consistently obtains better results than LocalSolver. For some instances, the difference can be as large as 21%. The runtimes of the heuristic are also a fraction of the allowable runtime that is given to LocalSolver. It needs to be noted, however, that LocalSolver uses 12 threads to solve the mathematical model, while the serial heuristic only uses one. A comparison between the runtimes of the heuristic that is implemented in parallel is a fairer comparison. On average, the heuristic delivers solutions with objective function values that are 12.4% lower compared to the solutions obtained by LocalSolver, in 0.06% of the runtime. The remaining, larger instances show consistent results within reasonable runtimes. The instances that are more difficult to solve, such as instance I_{18}, have a higher runtime because there are more construction iterations that yield infeasible solutions. This means that there are more construction iterations needed in total in order to reach the stopping criterion. If the instance becomes larger as well, that is, there are more variables, the runtime increases as well. This is because each construction cycle takes longer to complete. However, this increase in runtime is not as significant as the previously discussed increase. Furthermore, by comparing the runtimes between the serial and the parallel implementation we can see that the runtime can be decreased with a factor of up to 5.11 for some instances. On average, the runtimes decrease by a factor of 2.6. It is likely possible to further optimize the code to make the parallel implementation more scalable to the number of threads.

7 | SERVICE ANALYSIS

A comparison between the DRFS and a TFS is previously discussed by Galarza Montenegro et al. [22]. It is found that the DRFS performs significantly better than its traditional counterpart. Since the FSMS is an extension of the DRFS, we can expect similar results. Furthermore, in the FSMS fewer buses are needed to serve the same demand because the buses now return to the start to be reused. Moreover, passengers without a reservation are guaranteed a maximum waiting time at the mandatory stops.

For this reason we opt not to include a comparison with a traditional service. The influence of the different instance parameters on the service quality and on the runtime are discussed instead.

Several experiments are conducted in which four instances are solved with the heuristic. Each set of instances 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. Each set of instances is a subset of the instances shown in Table C1. The instance parameters that are discussed are: the number of passengers per hour, the number of buses, the number of mandatory stops, the number of optional stops per cluster, the bus capacity and the maximum allowable headway at mandatory stops D^{f}. The results are presented in Table E1 in Appendix E.

We further discuss the influence of the passengers without a reservation on the operation of the FSMS. The demand for transportation of these passengers is stochastic in nature. Therefore we limit ourselves to a descriptive analysis. A more in-depth analysis would require much more research, which we consider outside of the scope of this study.

7.1 | Number of passenger requests per hour

Out of all of the instance parameters, the number of passenger requests per hour affects the objective function value the most. As expected, the difference in departure times and the difference in arrival times increase the most. When there are more requests and the vehicle fleet remains the same, it is more difficult to provide a customized service to each passenger, that is, a service more reminiscent of a taxi service where passengers picked up or dropped off at their desired times. More requests consequently also leads to higher travel times and walking times because the buses need to drive longer routes. However, the increase in all metrics diminishes as more passengers are served. After a certain point, the passengers are more easily serviced in groups, which makes the service more efficient. The runtime increases exponentially with the number of passengers. This is because infeasible solutions are produced more often when there are more passenger requests to satisfy. This increases the total number of construction iterations and thus also the runtime. On top of this, each construction iteration takes longer as well because the number of possible assignments increases.

7.2 | Number of buses

It is clear that the objective function value decreases when more buses are utilized. The decrease is linear but seems to be less significant when a larger number of buses is used. All performance metrics seem to decrease with the use of more buses. However, the most significant decreases are the difference in arrival times and the difference in departure times. The second most significant decrease is the walking time. This is expected as more buses allow for a more tailored service for each passenger; more available buses allow for more customized routes and timetables. The runtime decreases exponentially with the use of more buses. This is likely due to the fact that it becomes easier to find feasible solutions and in turn, the heuristic needs less construction iterations to reach the stopping criterion.

7.3 | Number of mandatory stops

The objective function value increases linearly with the number of mandatory stops. Clearly, the biggest performance metric increase is the travel time of the passengers. This is because the travel distance to the destination automatically increases when there are more mandatory stops. The difference in arrival times and the difference in departure times are the second most significant increases. If the overall travel time of the buses increases, the buses need more time to return to the first mandatory stop for their next trip and thus have less time to pick up or drop off passengers at their desired times. The walking time slightly increases as well. This is likely because the algorithm prioritizes the efficiency of the routes over the bus stop assignment in order to decrease the objective function value further. The runtime increases exponentially. The longer travel times of the buses make it more difficult to satisfy the headway constraints and other scheduling constraints, which leads to more infeasible solutions and thus a longer runtime.

7.4 | Number of optional stops per cluster

An increasing number of optional stops per cluster has a small effect on the objective function value. The objective decreases slightly when more optional stops are used. The decrease diminishes as more optional stops are utilized. Evidently, the walking of the passengers decreases the most because passengers have more options for a departure stop. However, the trade-off for the decreased walking time is an increase in onboard travel time because the route becomes longer in order to visit more optional stops. On one hand, the longer travel times of the buses lead to an increase in the difference of arrival times because it becomes

more difficult to schedule their arrivals at the destination. On the other hand, the departure time difference decreases since it becomes easier to schedule the departure of the buses according to the passengers' needs because there are more possibilities for departure stops. There is a slight increase in runtime when more optional stops are used. This is because each construction iteration takes slightly longer to complete since there are more possibilities for the passenger-stop assignment and the routing of the buses.

7.5 | Vehicle capacity

The bus capacity has a smaller influence on the service quality. A larger capacity improves the objective function value slightly at first. As the capacity increases further, the objective function worsens slightly again. This can be explained as follows. When there is more capacity available, it becomes possible to find better solutions because there are more options for assigning passengers to buses. However, when the capacity continues to increase, the increasing number of options for bus assignments leads to a higher probability of getting trapped in a local minima during the search. This, in turn, can result in slightly worse solutions. The walking times increase slightly. This is because with higher vehicle capacities, passengers are able to be grouped more in order to decrease the other performance metrics. For the same reason, the arrival times difference increases as well. The difference in departure times decreases the most when the capacity is higher. Buses with a low capacity can only pick up a limited number of passengers, which means that passengers are often not assigned to a bus that has similar departure times compared to their desired departure time. When the capacity is higher, these assignments are more often possible. The travel times of the passengers are virtually unaffected, although they decrease very slightly with a higher capacity. This is likely the trade-off with the increase in walking times. The runtime seems to decrease slightly when the capacity is higher. The higher capacity makes it easier to find feasible solutions since the constraints are less tight. This leads to a lower number of construction iterations and thus to shorter runtimes.

7.6 | Maximum headway at mandatory stops D^f

The maximum headway at mandatory stops D^f has little to no influence on the objectives. When D^f is too low, the difference in arrival times increases because the stricter scheduling constraints make it more difficult to schedule the arrival of the buses. The other performance metrics decrease slightly to compensate for this increase. The runtime increases when D^f is lower because the stricter scheduling constraints make it more difficult to find feasible solutions.

7.7 | Impact of passengers without a reservation

We assume that passengers communicate their needs by making a transportation request. The quality of service for passengers who do not make a request cannot be measured directly since we do not know what their demands are. Their demand for transportation is more stochastic in nature. However, we can indirectly estimate the service quality of these passengers.

The heuristic optimizes the objective function value, which consists of minimizing the onboard travel time of passengers with a reservation. This, in turn, reduces the travel time of the buses, which also decreases the onboard travel time of passengers without reservation. Therefore, having more buses available, that is, a larger fleet size, indirectly increases the service quality of passengers without a reservation. Furthermore, when there are less passenger requests per hour, the onboard travel times of passengers with a reservation decrease as well. This implies that passengers without a reservation also benefit when there are less requests for transportation, since the bus travel times will decrease as well.

In the FSMS, bus departures must have a maximum headway D^f. This means that a bus cannot depart more than D^f seconds after the previous bus departure at any mandatory stop. However, departing earlier than D^f seconds after the previous bus departure is allowed. This, in turn, imposes a maximum waiting time for passengers without a reservation who are waiting to board a bus at a mandatory stop. Therefore, by decreasing the maximum headway D^f, we decrease the maximum waiting time and consequently increase the service quality of passengers without a reservation. As it was stated in the previous section, decreasing D^f does not influence the objective function value, that is, the service quality of the passengers with a reservation, by much. Only when D^f is too low, the runtime of the heuristic increases significantly.

Furthermore, passengers without a reservation who are waiting for a bus at a mandatory stop that is closer to the destination are more likely to wait longer for a bus when compared to passengers waiting in previous mandatory stops. The buses are more likely to be at capacity in the later mandatory stops. This means that passengers without a reservation might not be able to board the bus but have to wait for the next departure instead. As a trade-off, these passengers spend less time onboard the bus, since these mandatory stops are closer to the destination.

8 | CONCLUSION

The FSMS has been introduced in this article. The FSMS works with two types of bus stops: optional stops are only visited when there is demand for transportation, while mandatory stops need to be visited by a bus within a certain time frame. The FSMS is a semi-flexible DRTS that incorporates positive characteristics of both traditional transport services and fully flexible DRTS. On one hand, the service has flexibility in selecting which of the clustered optional bus stops are visited, based on online passenger requests. On the other hand, there is predictability in the mandatory bus stops. If online requests are not made, it is still possible to catch a bus in a mandatory bus stop. This will likely improve service quality.

In order to optimize the performance of the FSMS, a new type of metaheuristic framework is developed, namely the parameter space search (PSS) heuristic. In the PSS heuristic, solutions are constructed in a semi-random greedy manner in each construction iteration of the heuristic. In each construction iteration, the passenger-stop assignment is determined by making use of the pilot method framework. Randomized construction parameters create a balance between random and greedy constructions. This leads to a high variability in the solutions that are generated, which allows the heuristic to find feasible solutions on more strictly constrained instances. The heuristic uses local search, more specifically SA, to find the best parameter values in the parameter space of the underlying constructive heuristic. The parameters of the SA local search are fine-tuned to obtain good results for any kind of instance. This approach yields high quality results in short runtimes for 42 different instances. The first 15 instances are solved with the commercial optimization solver LocalSolver (LS) with a runtime of one hour. The solutions obtained with LS are used as benchmarks to assess the quality of the solution. It is found that the PSS heuristic yields 12.42% better results on average, within a few seconds, when compared to solutions obtained by LS. The remaining larger instances are typically solved within 2 min.

The influence of different instance parameters on the service quality and the runtime of the optimization of the service are discussed as well. Instance parameters such as passenger density, fleet size, number of stops, vehicle capacity and maximum allowable headway at the mandatory stops are considered. When more passengers make a request during the planning horizon, the service quality worsens as well. This is because passengers are forced to be grouped more, making the service less tailored to each customer. Following this logic, a larger fleet size improves the service quality. Using more optional stops also improves the service quality with a small increase in runtime. This is expected since more optional stops increase the level of flexibility in the service. More mandatory stops decrease the service quality because the main route becomes larger, making onboard travel times longer and increasing the time between bus trips. The vehicle capacity and the maximum headway have a limited influence on the service quality. However, a very small headway can result in large objective values. Therefore, headways larger than 15 min are recommended. Obviously the runtime increases with the size of the instances. However, it can be concluded that the runtime is influenced the most by how difficult an instance is, that is, how difficult it is to find feasible solutions. It is found that the passenger density, followed by the fleet size, have the largest influence on both the runtime and the service quality.

Further research could focus on optimizing the service in real-time. This will make the problem considerably more complex because it needs to modify routes and timetables to accommodate new requests, while still satisfying all constraints and trying to provide the best service. The heuristic presented in this work can be used as a starting point for the solution method of such a service. Quantifying how well the FSMS can serve the passengers that do not make a request for transportation is an interesting next step as well.

ACKNOWLEDGMENTS

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

DATA AVAILABILITY STATEMENT

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

ORCID

Bryan David Galarza Montenegro https://orcid.org/0000-0002-1827-5331

REFERENCES

[1] N. Agatz, M. Hewitt, and B. W. Thomas, *"Make no little plans": Impactful research to solve the next generation of transportation problems*, Networks **77** (2021), no. 2, 269–286.

[2] M. J. Alonso-González, T. Liu, O. Cats, N. Van Oort, and S. Hoogendoorn, *The potential of demand-responsive transport as a complement to public transport: An assessment framework and an empirical evaluation*, Transp. Res. Rec. **2672** (2018), no. 8, 879–889.

[3] C. Ansótegui, M. Sellmann, and K. Tierney, "*A gender-based genetic algorithm for the automatic configuration of algorithms*," *Principles and practice of constraint programming-CP 2009*, I. P. Gent (ed.), Springer, Berlin, 2009, pp. 142–157.

[4] N. Azizi and S. Zolfaghari, *Adaptive temperature control for simulated annealing: A comparative study*, Comput. Oper. Res. **31** (2004), no. 14, 2439–2451.

[5] P. Balaprakash, M. Birattari, and T. Stützle, "*Improvement strategies for the F-race algorithm: Sampling design and iterative refinement*," *Hybrid metaheuristics HM 2007*, T. Bartz-Beielstein, M. J. B. Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels (eds.), Springer, Berlin, 2007, pp. 108–122.

[6] E. B. Barbosa and E. L. F. Senne, "*A heuristic for optimization of metaheuristics by means of statistical methods*," *6th Int. Conf. Oper. Res. Enterp. Syst.*, Science and Technology Publications, Setúbal, Portugal, 2017, pp. 203–210.

[7] G. Beirão and J. A. Sarsfield Cabral, *Understanding attitudes towards public transport and private car: A qualitative study*, Transp. Policy **14** (2007), no. 6, 478–489.

[8] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "*A racing algorithm for configuring metaheuristics*," *Proc. 4th Annu. Conf. Genet. Evol. Comput.*, Vol **2**, Citeseer, San Francisco, CA, 2002, pp. 11–18.

[9] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "*F-race and iterated F-race: An overview*," *Experimental methods for the analysis of optimization algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss (eds.), Springer, Berlin, 2010, pp. 311–336.

[10] N. Blandamour. LocalSolver vs Gurobi on the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), 2022.

[11] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, *The vehicle routing problem: State of the art classification and review*, Comput. Ind. Eng. **99** (2016), 300–313.

[12] L. P. Cáceres and T. Stützle, *Exploring variable neighborhood search for automatic algorithm configuration*, Electron. Notes Discrete Math. **58** (2017), 167–174.

[13] S. I. Chien, L. N. Spasovic, S. S. Elefsiniotis, and R. S. Chhonkar, *Evaluation of feeder bus systems with probabilistic time-varying demands and nonadditive time costs*, Transp. Res. Rec. **1760** (2001), no. 1, 47–55.

[14] F. Ciaffi, E. Cipriani, and M. Petrelli, *Feeder bus network design problem: A new metaheuristic procedure and real size applications*, Procedia Soc. Behav. Sci. **54** (2012), 798–807.

[15] G. Clarke and J. W. Wright, *Scheduling of vehicles from a central depot to a number of delivery points*, Oper. Res. **12** (1964), no. 4, 568–581.

[16] T. G. Crainic, F. Errico, F. Malucelli, and M. Nonato, *Designing the master schedule for demand-adaptive transit systems*, Ann. Oper. Res. **194** (2012), no. 1, 151–166.

[17] T. G. Crainic, F. Malucelli, M. Nonato, and F. Guertin, *Meta-heuristics for a class of demand-responsive transit systems*, INFORMS J. Comput. **17** (2005), no. 1, 10–24.

[18] DeLijn. Dial-a-bus a solution for your journey? 2021. https://www.delijn.be/en/belbus/?vertaling=true.

[19] A. Dixit, A. Mishra, and A. Shukla, "*Vehicle routing problem with time windows using meta-heuristic algorithms: A survey*," *Harmony search and nature inspired optimization algorithms*, N. Yadav, A. Yadav, J. C. Bansal, K. Deep, and J. H. Kim (eds.), Springer, Singapore, 2019, pp. 539–546.

[20] N. A. El-Sherbeny, *Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods*, J. King Saud Univ. Sci. **22** (2010), no. 3, 123–131.

[21] L. Fu and Q. Liu, *Real-time optimization model for dynamic scheduling of transit operations*, Transp. Res. Rec. **1** (2003), no. 1857, 48–55.

[22] B. D. Galarza Montenegro, K. Sörensen, and P. Vansteenwegen, *A large neighborhood search algorithm to optimize a demand-responsive feeder service*, Transp. Res. Part C Emerg. Technol. **127** (2021), 103102.

[23] B. D. Galarza Montenegro, K. Sörensen, and P. Vansteenwegen, *A column generation algorithm for the demand-responsive feeder service with mandatory and optional, clustered bus-stops*, Networks **80** (2022), no. 3, 274–296.

[24] J. J. Grefenstette, *Optimization of control parameters for genetic algorithms*, IEEE Trans. Syst. Man Cybern. **16** (1986), no. 1, 122–128.

[25] R. Guo, W. Guan, W. Zhang, F. Meng, and Z. Zhang, *Customized bus routing problem with time window restrictions: model and case study*, Transp. A Transp. Sci. **15** (2019), no. 2, 1804–1824.

[26] S. Handy, L. Weston, and P. L. Mokhtarian, *Driving by choice or necessity?* Transp. Res. Part A Policy Pract. **39** (2005), 183–203.

[27] J. Hine and F. Mitchell, *Better for everyone? Travel experiences and transport exclusion*, Urban Stud. **38** (2001), no. 2, 319–332.

[28] F. Hutter, T. Stützle, K. Leyton-Brown, and H. H. Hoos. Paramils: An automatic algorithm configuration framework. arXiv print arXiv:1401, 2014.

[29] M. Kim and P. Schonfeld, *Integration of conventional and flexible bus services with timed transfers*, Transp. Res. B Methodol **68** (2014), no. 2014, 76–97.

[30] M. E. Kim and P. Schonfeld, *Integrating bus services with mixed fleets*, Transp. Res. B Methodol **55** (2013), 227–244.

[31] A. Lakatos, J. Tóth, and P. Mándoki, *Demand responsive transport service of 'dead-end villages' in interurban traffic*, Sustainability **12** (2020), no. 9, 3820.

[32] A. Lee and M. Savelsbergh, *An extended demand responsive connector*, EURO J. Transp. Logist. **6** (2017), no. 1, 25–50.

[33] X. Li. *Optimal design of demand-responsive feeder transit services*, Ph.D. thesis, Texas A&M University, 2009.

[34] X. Li and L. Quadrifoglio, *Feeder transit services: Choosing between fixed and demand responsive policy*, Transp. Res. Part C Emerg. Technol. **18** (2010), no. 5, 770–780.

[35] J. J. Lin and H. I. Wong, *Optimization of a feeder-bus route design by using a multiobjective programming approach*, Transp. Plan. Technol. **37** (2014), no. 5, 430–449.

[36] T. Liu and A. A. Ceder, *Analysis of a new public-transport-service concept: Customized bus in china*, Transp. Policy **39** (2015), 63–76.

[37] X. Lu, J. Yu, X. Yang, S. Pan, and N. Zou, *Flexible feeder transit route design to enhance service accessibility in urban area*, J. Adv. Transp. **50** (2015), no. 4, 507–521.

[38] O. Maron and A. W. Moore, *The racing algorithm: Model selection for lazy learners*, Artif. Intell. Rev. **11** (1997), 193–225.

[39] C. L. Martins and M. V. Pato, *Search strategies for the feeder bus network design problem*, Eur. J. Oper. Res. **106** (1998), 425–440.

[40] B. Mehran, Y. Yang, and S. Mishra, *Analytical models for comparing operational costs of regular bus and semi-flexible transit services*, Public Transp. **12** (2020), no. 1, 147–169.

[41] E. Melachrinoudis, A. B. Ilhan, and H. Min, *A dial-a-ride problem for client transportation in a health-care organization*, Comput. Oper. Res. **34** (2007), no. 3, 742–759.

[42] M. Mistretta, J. A. Goodwill, R. Gregg, and C. DeAnnuntis. *Best practices in transit service planning*, Final Report No. BD549-38. Technical report, Center for Urban Transportation Research for the Florida Department of Transportation, 2009.

[43] A. S. Mohaymany and A. Gholami, *Multimodal feeder network design problem: Ant colony optimization approach*, J. Transp. Eng. **136** (2010), no. 4, 323–331.

[44] A. Pratelli, M. Lupi, A. Farina, and C. Pratelli. *Comparing route deviation bus operation with respect to dial-a-ride service for a low-demand residential area* (7th Int. Conf. Data Anal.), 2018, pp. 151.
[45] F. Qiu, W. Li, and A. Haghani, *An exploration of the demand limit for flex-route as feeder transit services: a case study in Salt Lake City*, Public Transp. **7** (2015), no. 2, 259–276.
[46] L. Quadrifoglio and M. M. Dessouky, "*Mobility allowance shuttle transit (mast) services: formulation and simulation comparison with conventional fixed route bus services*," *Proc. 4th IASTED Int. Conf. Model. Simul. Optim.*, Acta Press, Calgary, AB, 2004, pp. 31–36.
[47] L. Quadrifoglio and X. Li, *A methodology to derive the critical demand density for designing and operating feeder transit services*, Transp. Res. B Methodol. **43** (2009), no. 10, 922–935.
[48] M. Resende and C. Ribeiro. *An introduction to GRASP* (XXXIX Simpósio Brasileiro de Pesquisa Operacional), Fortaleza, Brazil, 2007, pp. 3092–3227.
[49] P. Saeidizand. *Urban public transport in the 21st century*. Report, Advancing Public Transport, 2017.
[50] P. Shrivastava and M. O'Mahony, *A model for development of optimized feeder routes and coordinated schedules—A genetic algorithms approach*, Transp. Policy **13** (2006), 413–425.
[51] P. Shrivastava and M. O'Mahony, *Design of feeder route network using combined genetic algorithm and specialized repair heuristic*, J. Public Transp. **10** (2007), no. 2, 109–133.
[52] B. Sun, M. Wei, and S. Zhu, *Optimal design of demand-responsive feeder transit services with passengers' multiple time windows and satisfaction*, Future Internet **10** (2018), no. 3, 30.
[53] P. Vansteenwegen, L. Melis, D. Aktaş, B. D. Galarza Montenegro, F. Veiera, and K. Sörensen, *A survey on demand-responsive public bus systems*, Transp. Res. Part C Emerg. Technol. **137** (2022), 103573.
[54] S. Voß, A. Fink, and C. Duin, *Looking ahead with the pilot method*, Ann. Oper. Res. **136** (2005), no. 1, 285–302.
[55] L. Wang, S. Wirasinghe, L. Kattan, and S. Saidi, *Optimization of demand-responsive transit systems using zonal strategy*, Int. J. Urban Sci. **22** (2018), no. 3, 366–381.
[56] M. Wei, T. Liu, B. Sun, and B. Jing, *Optimal integrated model for feeder transit route design and frequency-setting problem with stop selection*, J. Adv. Transp. **2020** (2020), 1–12.
[57] N. Wilson, *Scheduling Algorithms for a Dial-a-ride System*, PB 201 808, Massachusetts Institute of Technology, Urban Systems Laboratory, Cambridge, MA, 1971.
[58] Z. Yuan, M. A. Montes de Oca, M. Birattari, and T. Stützle, *Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms*, Swarm Intell. **6** (2012), 49–75.
[59] M. Zheng, R. Zhou, S. Liu, F. Liu, and X. Guo, *Route design model of multiple feeder bus service based on existing bus lines*, J. Adv. Transp. **2020** (2020), 8853872.
[60] Y. Zheng, W. Li, and F. Qiu, *A methodology for choosing between route deviation and point deviation policies for flexible transit services*, J. Adv. Transp. **2018** (2018), 6292410.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: B. D. Galarza Montenegro, K. Sörensen, and P. Vansteenwegen, *A demand-responsive feeder service with a maximum headway at mandatory stops*, Networks. **83** (2024), 100–130. https://doi.org/10.1002/net.22185

APPENDIX A: LINEARIZED CONSTRAINTS OF THE MATHEMATICAL MODEL

Linear version of constraints (16):

$$d^{s}_{bti} - \left(1 - y_{pbti}\right)M \leq d_p \leq d^{s}_{bti} + \left(1 - y_{pbti}\right)M \qquad \forall\, i \in S,\ b \in B,\ t \in J,\ p \in P_2. \tag{A1}$$

Linear version of constraints (17):

$$d^{s}_{bt|F|-1} - \left(1 - y_{pbti}\right)M \leq a_p \leq d^{s}_{bt|F|-1} + \left(1 - y_{pbti}\right)M \qquad \forall\, b \in B,\ t \in J,\ p \in P_1. \tag{A2}$$

Linear version of constraints (20):

$$d^{s}_{bti} + T^{t}_{ij} - \left(1 - x_{btij}\right)M \leq d^{s}_{btj} \leq d^{s}_{bti} + T^{t}_{ij} + \left(1 - x_{btij}\right)M \qquad \forall\, i \in S_{|F|-1},\ j \in S_0,\ b \in B,\ t \in J. \tag{A3}$$

Linear version of constraints (24):

$$\sum_{p \in P} y_{pbti} \leq M \sum_{l \in S} x_{bitl} \qquad \forall\, i \in O,\ b \in B,\ t \in T. \tag{A4}$$

With M representing sufficiently large numbers.

APPENDIX B: PSEUDO-CODE OF THE CONSTRUCTION ALGORITHM

Algorithm 2. Construction algorithm outline

Determine optimal route BR visiting all stops
Determine travel time T^{r} to visit all mandatory stops
Sort passengers $p \in P$ in ascending T_p^{arr} and T_p^{dep}
Place passengers with a T_p^{arr} in queue Q^{a}, passengers with T_p^{dep} in queue Q^{d}
t_b^{d} = start time optimization $\forall b \in B$
it = 0
while Q^{a} *not empty* ***and*** Q^{d} *not empty* **do**
  $D_{bt}^{\mathrm{f}} = D^{\mathrm{f}} r_u^{\mathrm{d}}$
  Next bus b = earliest available bus
  Make route R^{b} visiting all mandatory stops.
  Timetable of bus b: $d_0 = t_b^{\mathrm{d}}$, $d_i = d_{i-1} + T^{\mathrm{t}}_{(i-1)i}$
  Number of passengers onboard bus b, $N_b^{\mathrm{p}} = 0$
  $T^{\mathrm{a}} = \min_{p \in Q^{\mathrm{a}}} T_p^{\mathrm{arr}}$, $T^{\mathrm{d}} = \min_{p \in Q^{\mathrm{d}}} T_p^{\mathrm{dep}} + T^{\mathrm{r}}$
  if $T^{\mathrm{a}} < T^{\mathrm{d}}$ ***and*** Q^{a} *not empty* **then**
    while $N_b^{\mathrm{p}} < C$ ***and*** Q^{a} *not empty* **do**
      s=`Best_Stop_a`$(p_1, b, t, r_u^{\mathrm{f}})$
      if $s \neq -1$ **then**
        Assign p_1 in Q^{a} to stop s and to bus b on trip t
        Remove p_1 from Q^{a}
        `Update_route`(BR, s, R^{b})
        `Update_timetable`$(m_{|F|-1}, p_1, D^{\mathrm{la}}, D^{\mathrm{ea}}, D_{bt}^{\mathrm{f}})$
        N_b^{p}++
      else
        Stop adding passengers
      end
    end
  end
  if $T^{\mathrm{a}} > T^{\mathrm{d}}$ ***and*** Q^{d} *not empty* **then**
    while Q^{d} *not empty* ***and*** $N_b^{\mathrm{p}} < C$ **do**
      s=`Best_Stop_d`$(p_2, b, t, T_{p_2}^{\mathrm{dep}}, D^{\mathrm{ld}}, r_u^{\mathrm{f}})$
      if $s \neq -1$ **then**
        Assign passenger p_2 to s and to bus b on trip t
        Remove p_2 from Q^{d}
        `Update_route`(BR, s, R^{b})
        `Update_timetable`$(s, p_2, D^{\mathrm{ld}}, D^{\mathrm{ed}}, D_{bt}^{\mathrm{f}})$
        N_b^{p}++
      else
        Stop adding passengers
      end
    end
  end
  if *bus b is empty* **then** Timetable of bus b: $d_0 = t_b^{\mathrm{d}} + D^{\mathrm{f}}$, $d_i = d_{i-1} + T^{\mathrm{t}}_{(i-1)i}$;
  else
    Calculate D^{sp} and D^{sf}
    for *all remaining passengers p* **do**
      Calculate T_p^{s} for passenger p
      for *all bus stops s already part of the route* **do**
        Calculate walking time T_{ps}^{w} for passenger p to stop s
        if $T_{ps}^{w} < r_u^{w} D^{w}$ ***and*** $|T_p^s| < r_u^{sp} D^{sp}$ ***and*** $|T_p^s| < r_u^{sf} D^{sf}$ **then**
          Assign passenger p to bus b on trip t
          Modify timetable with T_p^{s}
          Recalculate D^{sp} and D^{sf}
        end
      end
    end
  end
  Add a trip to bus b
  Update t_b^{d} and sort buses again
  it++
end

Algorithm 3. Function for choosing best departure stop for $p_1 \in Q^{\mathrm{a}}$

Function `Best_Stop_a(`p_1, b, t, r_u^{f}`)`**:**
  $BestCost = \infty$
  $BestStop = -1$
  for *all stops* $s \in S$ *within walking distance of passenger* p_1 **do**
    cost=walking time of p_1 to s
    if s *is not part of the route already* **then**
      Determine the extra time T^{extra} it will take to go to s
      if T^{extra} *is too large to maintain departure time interval constraints* **then** cost=∞;
      else cost=cost+T^{extra};
    end
    Determine departure time T^{ds} at stop s
    if T^{ds} *makes arrival constraints infeasible* ***or*** $r_u^{\mathrm{f}} < R^{\mathrm{f}}$ **then**
      $BestCost = \infty$
    end
    if $BestCost > cost$ **then**
      $BestCost = cost$
      $BestStop = s$
    end
  end
  if $BestCost = \infty$ **then**
    Adding p_1 to bus b is infeasible
    return -1
  else
    return *BestStop*
  end

Algorithm 4. Function for updating a timetable

Function `Update_timetable(`s, p, D^{l}, D^{e}, D_{bt}^{f}`)`**:**
  $DT = T^{\mathrm{dep}}$ or T^{arr} of all passengers boarding or alighting in s
  M^{t}= median(DT)
  $E^{\mathrm{t}} = \min(DT)$
  $L^{\mathrm{t}} = \max(DT)$
  if $M^{\mathrm{t}} > \min\left(E^{\mathrm{t}} + D^{\mathrm{l}}, L^{\mathrm{t}}\right)$ **then**
    $M^{\mathrm{t}} = \min\left(E^{\mathrm{t}} + D^{\mathrm{l}}, L^{\mathrm{t}}\right)$
  else if $M^{\mathrm{t}} < \max\left(L^{\mathrm{t}} - D^{\mathrm{e}}, E^{\mathrm{t}}\right)$ **then**
    $M^{\mathrm{t}} = \max\left(L^{\mathrm{t}} - D^{\mathrm{e}}, E^{\mathrm{t}}\right)$
  end
  Calculate mrax difference M^{x} in departure times in mandatory stops
  if $M^{\mathrm{x}} > 0$ **then** $M^{\mathrm{t}} = M^{\mathrm{t}} - (D_{bt}^{\mathrm{f}} - M^{\mathrm{x}})$;
  Departure time at stop s is M^{t}
  Adjust other departure times accordingly

Algorithm 5. Function for updating a route

Function `Update_route(`BR, s, R^{b}`)`**:**
  if $s \notin R^{\mathrm{b}}$ **then**
    Determine closest stop $s_b \in R^{\mathrm{b}}$ that is visited before s in BR
    Insert s after s_b in R^{b}
  end

Algorithm 6. Function for choosing best departure stop for $p_2 \in Q^{\mathrm{d}}$

Function `Best_Stop_d`(p_2, b, t, $T^{dep}_{p_2}$, D^{ld}, r^{f}_u)**:**
- $BestCost = \infty$
- $BestStop = -1$
- **for** *all stops* $s \in S$ *within walking distance of passenger* p_2 **do**
  - **if** s *is inserted in* R^{b} *before the previously inserted stop in the route* **then**
    - cost=∞
  - **else**
    - cost=walking time of p_2 to s
    - Determine time of arrival T^{as} at s
    - **if** $T^{as} > T^{dep}_{p_2} - D^{ld}$ **then**
      - cost=∞
    - **else**
      - **if** s *is not part of the route already* **then**
        - Determine the extra time T^{extra} it will take to go to s
        - **if** T^{extra} *is too large to maintain the departure time interval constraints* ***or*** $r^{\mathrm{f}}_u < R^{\mathrm{f}}$ **then**
          - cost=∞
        - **else**
          - Determine best feasible (T^{dep} constraints) departure time $T^{\mathrm{d}s}$ at s
          - Determine difference $\Delta = T^{\mathrm{d}s} - T^{\mathrm{as}}$ of arrival time and a feasible departure time
          - cost=cost+T^{extra}+Δ
        - **end**
      - **else**
        - Determine latest feasible departure time T^{lf} at s
        - **if** $T^{as} > T^{lf}$ ***or*** $r^{\mathrm{f}}_u < R^{\mathrm{f}}$ **then**
          - cost=∞
        - **else**
          - Determine best feasible (T^{dep} constraints and departure time interval constraints) departure time $T^{\mathrm{d}s}$ at s
          - Determine difference $\Delta = T^{\mathrm{d}s} - T^{\mathrm{as}}$ of arrival time and a feasible departure time
          - cost=cost+T^{extra}+Δ
        - **end**
      - **end**
      - **if** $BestCost > cost$ **then**
        - $BestCost = cost$
        - $BestStop = s$
      - **end**
    - **end**
  - **end**
- **end**
- **if** $BestCost = \infty$ **then**
  - Adding p_2 to bus b is infeasible
  - **return** -1
- **else**
  - **return** *BestStop*
- **end**

APPENDIX C: INSTANCES OF THE EXPERIMENTS

TABLE C1 List of test instances for the experiments.

Instance	$\|B\|$	$\|F\|$	K	$\|S\|$	$\|P\|$	D^f (min)	C
I_1	**2**	**4**	**5**	**19**	**10**	**20**	**20**
I_2	**3**	**4**	**5**	**19**	**10**	**20**	**20**
I_3	2	4	5	19	12	20	20
I_4	3	4	5	19	12	20	20
I_5	2	4	5	19	16	20	20
I_6	3	4	5	19	16	20	20
I_7	3	4	5	19	18	20	20
I_8	4	4	5	19	18	20	20
I_9	4	4	5	19	20	20	20
I_{10}	3	5	5	25	10	20	20
I_{11}	**3**	**5**	**5**	**25**	**16**	**20**	**20**
I_{12}	4	5	5	25	20	20	20
I_{13}	3	6	5	31	10	20	20
I_{14}	4	6	5	31	16	20	20
I_{15}	**4**	**6**	**5**	**31**	**20**	**20**	**20**
I_{16}	**3**	**4**	**5**	**19**	**20**	**20**	**20**
I_{17}	**3**	**5**	**5**	**25**	**20**	**20**	**20**
I_{18}	**5**	**6**	**5**	**31**	**48**	**20**	**20**
I_{19}	**6**	**6**	**5**	**31**	**48**	**20**	**20**
I_{20}	**7**	**6**	**5**	**31**	**48**	**20**	**20**
I_{21}	8	6	5	31	48	20	20
I_{22}	12	6	8	46	20	20	20
I_{23}	12	6	8	46	48	20	20
I_{24}	**10**	**5**	**8**	**37**	**90**	**20**	**20**
I_{25}	**10**	**6**	**8**	**46**	**90**	**20**	**20**
I_{26}	**10**	**7**	**8**	**55**	**90**	**20**	**20**
I_{27}	12	6	8	46	90	20	20
I_{28}	**11**	**6**	**5**	**31**	**162**	**20**	**20**
I_{29}	**12**	**6**	**5**	**31**	**162**	**20**	**20**
I_{30}	**12**	**6**	**8**	**46**	**162**	**20**	**20**
I_{31}	12	5	8	37	162	20	20
I_{32}	12	7	8	55	162	20	20
I_{33}	12	8	8	64	162	20	20
I_{34}	12	6	3	21	162	20	20
I_{35}	12	6	10	56	162	20	20
I_{36}	6	5	8	37	90	20	5
I_{37}	6	5	8	37	90	20	10
I_{38}	6	5	8	37	90	20	20
I_{39}	6	5	8	37	90	20	30
I_{40}	6	5	8	37	90	15	20
I_{41}	6	5	8	37	90	25	20
I_{42}	6	5	8	37	90	30	20

Note: The instances in bold represent a subset which is used to fine-tune the local search parameters of the PSS heuristic.

APPENDIX D: RESULTS OF THE EXPERIMENTS

This table shows the min and mean values of the objective function from 10 runs. The minimum and mean gaps, that is, the gap between the values obtained by LocalSolver and the observed values, are also presented. The runtimes of both the serial and parallel implementation of the heuristic are shown as well. The 16th row shows the mean values of instances I_1 to I_{15}.

TABLE D1 Computational results of the experiments on all instances.

	Objective function value (s)					Runtime (s)		
Instance	**Heuristic mean**	**Heuristic min**	**Local solver**	**Gap mean**	**Gap min**	**Local solver**	**Heuristic parallel**	**Heuristic serial**
I_1	4278	4242	4420	−3.22%	−4.03%	3600	1.93	3.53
I_2	3762	3760	4055	−7.22%	−7.26%	3600	0.75	1.76
I_3	5491	5414	5993	−8.38%	−9.66%	3600	2.33	3.71
I_4	4758	4689	5858	−18.8%	−20.0%	3600	1.48	2.88
I_5	5597	5597	6951	−20.0%	−20.0%	3600	5.82	7.14
I_6	5305	5243	5952	−11.9%	−11.9%	3600	1.35	3.43
I_7	6756	6684	8456	−20.1%	−21.0%	3600	3.04	5.15
I_8	6643	6384	7829	−15.2%	−18.5%	3600	0.82	2.28
I_9	7499	7332	8546	−12.3%	−14.2%	3600	0.80	2.39
I_{10}	4597	4597	5079	−9.51%	−9.51%	3600	0.90	2.04
I_{11}	6708	6655	8203	−18.2%	−18.9%	3600	3.53	4.31
I_{12}	9417	9212	10 897	−13.6%	−15.5%	3600	1.59	3.42
I_{13}	5409	5409	5641	−4.12%	−4.12%	3600	2.20	3.49
I_{14}	7813	7672	8884	−12.1%	−13.6%	3600	3.30	4.50
I_{15}	10 762	10 762	12 412	−13.3%	−13.3%	3600	1.90	3.17
$\overline{\mu}_{I_1-I_{15}}$				−12.4%	−13.4%	3600	2.12	3.55
I_{16}	7958	7817					2.45	3.81
I_{17}	10 133	10 093					7.10	6.14
I_{18}	23 244	22 598					63.7	120
I_{19}	21 843	21 543					17.3	37.0
I_{20}	21 092	20 641					6.09	13.9
I_{21}	20 963	20 601					2.63	8.93
I_{22}	10 580	10 493					0.61	2.07
I_{23}	20 451	19 850					1.92	8.47
I_{24}	39 332	38 511					5.08	17.2
I_{25}	45 375	44 138					7.46	23.4
I_{26}	52 698	51 736					12.5	35.8
I_{27}	46 315	45 662					4.50	17.6
I_{28}	89 538	88 258					21.8	53.0
I_{29}	88 840	87 008					15.3	34.8
I_{30}	87 844	86 904					17.0	42.3
I_{31}	75 270	73 346					9.20	33.3
I_{32}	99 643	98 145					55.3	88.9
I_{33}	112 392	110 486					126	204
I_{34}	90 195	87 817					14.6	31.1
I_{35}	89 018	86 742					16.4	42.0
I_{36}	40 535	39 912					83.8	323
I_{37}	40 129	39 328					52.7	201
I_{38}	39 900	39 518					40.9	158
I_{39}	40 756	39 950					43.7	160
I_{40}	40 395	39 608					83.3	224
I_{41}	40 166	39 567					30.8	153
I_{42}	40 158	39 567					29.1	150

APPENDIX E: SERVICE ANALYSIS

In this table the objective function value per passenger is shown in column three. The average travel time and walking time per passenger is given in columns five and six respectively. The average difference in desired and actual arrival time per passenger and the average difference in desired and actual departure time per passenger are given in the last two columns. Since each instance is run 10 times, we consider the best observed solution for the objective function value and the service quality metrics. The runtime in this table is the average runtime of the 10 runs.

TABLE E1 Results for the service analysis of the FSMS.

Inst.	Parameter	Objective value (s)	Runtime (s)	Travel time (s)	Walking time (s)	$\lvert T_p^{arr} - a_p \rvert$ (s)	$\lvert T_p^{dep} - d_p \rvert$ (s)
	Passengers per hour						
I_{22}	5	293	0.61	602	265	72	12
I_{23}	12	414	1.92	788	359	119	67
I_{27}	23	507	4.50	921	425	238	128
I_{30}	41	536	17.0	965	474	224	138
	Buses						
I_{18}	5	471	63.7	825	443	207	120
I_{19}	6	449	17.3	817	402	177	100
I_{20}	7	430	6.09	795	415	150	60
I_{21}	8	429	2.63	793	390	167	53
	Mandatory stops						
I_{31}	5	453	9.20	733	473	200	112
I_{30}	6	536	17.0	965	474	224	138
I_{32}	7	606	55.3	1144	482	228	176
I_{33}	8	682	126	1368	495	238	142
	Optional stops per cluster						
I_{34}	3	542	14.6	958	520	186	125
I_{29}	5	537	15.3	985	478	207	118
I_{30}	8	536	17.0	965	474	224	138
I_{35}	10	535	16.3	951	484	243	116
	Bus capacity						
I_{36}	5	443	83.8	741	417	208	158
I_{37}	10	437	52.7	697	450	240	117
I_{38}	20	439	40.9	695	444	254	104
I_{39}	30	444	44.6	701	444	243	118
	D^f (min)						
I_{40}	15	440	83.3	690	443	260	104
I_{38}	20	439	40.9	695	444	254	104
I_{41}	25	439	35.3	695	444	254	104
I_{42}	30	439	35.3	695	444	254	104