Operations Research Forum (2024) 5:25
https://doi.org/10.1007/s43069-024-00296-x

RESEARCH

Integrated Bus Timetabling and Scheduling with a Mutation-Based Evolutionary Scheme Maximizing Headway Quality and Connections

Lucas Mertens[1] · **Bastian Amberg**[1] · **Natalia Kliewer**[1]

Received: 31 January 2023 / Accepted: 19 January 2024

Abstract

Public transport planning is a multi-level process that includes various complex tasks. These tasks are traditionally executed sequentially, and the result of each task serves as input for consecutive tasks. A simultaneous integrated consideration of multiple tasks may lead to an overall improved solution, but further increase the complexity of already hard-to-solve planning problems. This work focuses on timetabling and vehicle scheduling and evaluates synergies from the integrated optimization. We investigate an exact sequential, exact integrated, and heuristic approach to solve the combined problem for large public transport networks considering the interlining of vehicles, multiple vehicle types, or multiple depots while additionally aiming to maximize regular "clock-faced" headways and transfer connections. Compared to sequential optimization, an integrated approach significantly reduces nominal and operational costs while maintaining high service quality. However, an exact integrated approach is only able to compute solutions for problems of limited size in a reasonable time. We propose an adaptive modular evolutionary extendable scheme that effectively balances computational efficiency and solution quality. By utilizing various problem-specific mutation operators and adaptively applying them based on their impact, the heuristic can compute high-quality solutions for large real-world-inspired public transport networks in a reasonable time while considering short connecting times between lines and regular clock-faced headways.

This article is part of the Topical Collection on *Public Transport Optimization: From Theory to Practice*

✉ Lucas Mertens
lucas.mertens@fu-berlin.de

Bastian Amberg
bastian.amberg@fu-berlin.de

Natalia Kliewer
natalia.kliewer@fu-berlin.de

[1] Department of Information Systems, Freie Universität Berlin, Garystr. 21, Berlin 14195, Germany

Published online: 15 March 2024

Keywords Public bus transport · Timetabling · Vehicle scheduling · Headways · Integrated scheduling

1 Introduction

Public transport planning is a multi-level planning process traditionally executed as a sequence of planning tasks [1]. Each task depends on the results of its predecessor, covers a different time horizon, and advances for shorter periods. Strategic planning covers the longest period of time and is separated into tasks of network design and line planning. Following these steps, tactical planning is divided into frequency setting and timetabling. For repeating short periods, operational planning includes vehicle scheduling, crew scheduling, and crew rostering, among others. This study focuses on the integrated optimization of the *timetabling problem* (TT) and the *vehicle scheduling problem* (VSP) to elaborate on synergies between the subsequent planning steps. Given various frequency and service requirements, solving the TT aims to define a schedule of trips for each line, that is, for each predefined route in the transport network. These service trips define departure and arrival times for each stop and form the timetable. A conflict of interest shapes the TT: on the one hand, solving the TT aims to maximize passenger satisfaction, for example, by maximizing the number of possible timely transfers between lines, or scheduling trips as regularly as possible. On the other hand, the anticipated costs of resources needed to serve this timetable must be reasonable. A common approach addressing this conflict is to assume a fixed number of service trips or a fixed frequency [1]. However, it is not always possible to determine a reasonable number of service trips or the frequency of departures prior to solving the TT. Hence, given a frequency range or without a given frequency, the quantity of service trips should be considered in optimization. This leads to increased complexity and demands more sophisticated solution approaches.

The VSP aims to generate a cost-minimal vehicle schedule and assign service trips to buses. Hence, the timetable's structure and the quantity of service trips greatly influence the VSP. Individually solving the TT first and the VSP second has been extensively researched, and well-performing approaches can be applied to large real-world problems in sequential public transport planning. However, a sequential approach might not lead to the best overall solution. Given the TT's requirements, generally, multiple different timetables of equal service quality can be computed. Solving the VSP for any of these timetables might lead to significantly different nominal and operational costs. Solving the *integrated timetabling and vehicle scheduling problem* (TTVSP) aims to simultaneously compute the timetable and vehicle schedule, leading to the minimal overall costs while complying with all service requirements.

Figure 1 illustrates an exemplary *public transport network* (PTN) serving as input for solving the TTVSP. The visualized PTN comprises four lines covering a total of 14 stops (blue circles). The first line (blue) crosses the second line (yellow) at its final destination and runs parallel to the third line (orange) for three stops after they intersect. Line four (purple) crosses both lines one and three at their second to last stop. At specific points in the PTN, service requirements are defined for the timetable (green symbols indicate virtual demand checkpoints). These requirements comprise

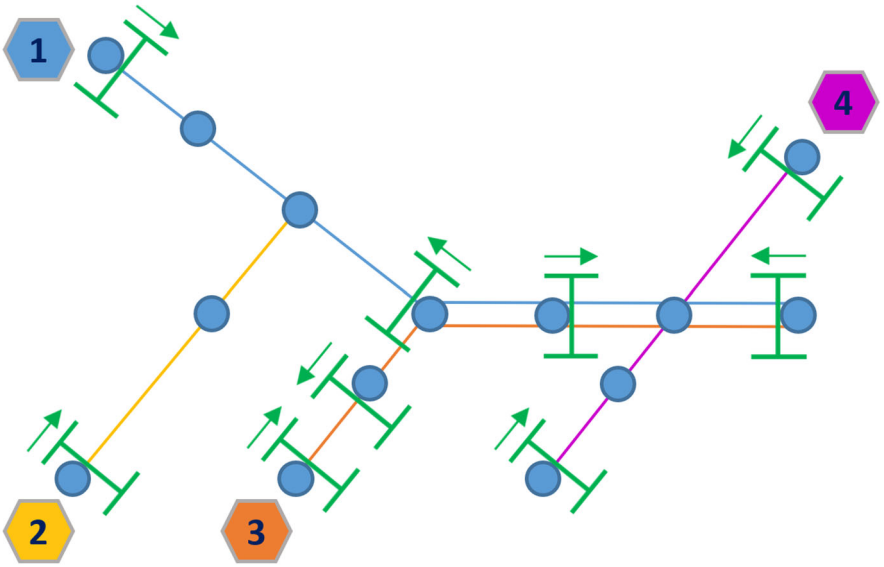


Fig. 1 A public transport network with 4 lines, 14 stop points, and virtual demand checkpoints

the least amount of passengers that must be transported for each hour of service time, as well as the range of minimum required and maximally allowed headways between two consecutive departures. A headway is measured at a specific stop and refers to the time interval between two successive departures of vehicles heading in the same direction to a common next stop, regardless of whether they are associated with the same line. Service requirements are measured right after a specific stop for each vehicle traveling in the indicated direction (green arrows). With the increasing complexity and size of the underlying PTN and the flexibility of the timetable, an exact approach for solving the TTVSP is incapable of computing an optimal solution in reasonable computation time for large-scale problems [2]. In this work, we increase the scope of the TTVSP by considering additional quality criteria perceived as desirable for public transport planners. With these criteria, we aim to maximize the number of headways of a desired quality and the number of transfer connections between lines, since these are important factors that passengers appreciate and consider when choosing routes for their journeys. Headways are considered good if they are regular and clock-faced, indicating a consistent time interval between successive trips and departure intervals as a factor of 60, respectively. Transfer connections (hereafter referred to as connections) are defined at intersections of lines. The time between the arrival and departure of the intersecting lines is desired to be at an appropriate interval to enable timely passenger transfers, therefore increasing the synchronization of lines. Neither of these additional quality criteria can be efficiently considered, even for small real-world-inspired instances, when solving the resulting *integrated timetabling and vehicle scheduling problem with good headways and maximizing connections* (TTVSP-HC) exactly [3].

This paper motivates solving the TTVSP and describes the scope and benefit of solving the TTVSP-HC. Based on the aforementioned difficulty in finding solutions to this problem, we propose a population-based heuristic that leverages heuristic techniques to generate good, feasible solutions efficiently. An *adaptive modular evolutionary extendable scheme* (AMEES) is applied to compute cost-minimizing timetables and vehicle schedules, considering defined timetable requirements, good headways, and connections. The scheme utilizes a generic solution manager coupled with a variety of TTVSP-HC-specific *Mutation Operators* (MOs). Each MO serves a different purpose and applies small mutations to improve the overall solution gradually. We evaluate the applicability of the proposed heuristic scheme by solving different real-world-inspired problem instances and comparing the results to an exact sequential and exact integrated approach.

This paper is organized as follows: Section 2 gives a brief overview of the state-of-the-art approaches for solving the TT, VSP, and TTVSP. We give a detailed problem description and propose a mathematical model for solving the TTVSP and TTVSP-HC exactly in Section 3. Section 4 gives a detailed overview of the newly developed heuristic scheme. Section 5 evaluates the benefit of optimizing the TTVSP and presents the computational results for both the exact and heuristic approaches to solve the TTVSP and TTVSP-HC. Finally, Section 6 concludes the paper, provides managerial insights, and gives an outlook to future research.

2 Literature Review

The primary concern of a bus company is how to allocate its limited resources to meet the passengers' trip demands efficiently. The TT greatly influences passenger satisfaction and the VSP the bus company's costs for operating the desired services. Solving the TT results in a set of trips represented in a timetable. Depending on the periodicity, consideration of vehicle types, vehicle capacities, and synchronization of lines, the TT is considered NP-hard for most realistic cases [2, 4]. Research considering the TT can be mainly divided by the following overall objectives: meeting specific demand patterns, minimizing waiting times for passengers, or maximizing synchronization. For example, [5, 6] propose models to create timetables with balanced passenger loads to meet highly variable passenger demand patterns during the day with short planning periods. Considering heterogeneous passenger fluctuations, [7] proposes a quadratic semi-assignment formulation to minimize passengers' waiting times at transfer stops. Numerous publications (e.g., [8, 9]) extend the early approach [7] to solve larger models in reasonable computation time. If the passenger flow is not known in advance, transfer synchronization of different lines can be maximized alternatively (e.g., [10, 11]). Apart from these three major objectives, several publications focus on achieving multiple objectives simultaneously (e.g., [12, 13]) or aim for a more robust timetable (e.g., [14]). For a detailed review of approaches to solve the TT, we refer the reader to [2].

The VSP aims to generate a cost-minimal vehicle schedule. Service trips are assigned to vehicles and complemented by deadhead trips without passengers to plan pull-outs from depots, pull-ins to depots, and trips between different lines. Consider-

ing only one depot and a homogeneous vehicle fleet, the *single depot, single vehicle type VSP* (SDVSP) corresponds to a minimum cost flow problem [15], which can be solved in polynomial time. Paper [16] reported the first approach to solve the SDVSP. The author formulates a network flow problem and uses a labeling algorithm to solve instances of up to 319 trips. However, while labeling the trips in order of departure times, the approach does not allow deadhead trips between two service journeys. Paper [17] advances this idea and includes deadhead trips. Since these early approaches, the structure of the underlying VSP has evolved substantially. Considering multiple vehicle types or multiple depots increases the problem's complexity significantly. The so-called multi-depot multi-vehicle type VSP is classified as NP-Hard [18]. For a detailed overview of vehicle scheduling models, we refer the reader to [19]. Even though solution approaches for real-world VSP and TT have been researched extensively and are applicable for large-scale problems in reasonable computation time, sequentially solving the TT first and the VSP second might not lead to the best overall solution. The next section gives an overview of state-of-the-art approaches for solving the TTVSP.

2.1 Integrated Timetabling and Vehicle Scheduling

Due to the high dependency of the VSP on the solution computed for the TT, an integrated approach considering both planning steps seems necessary. To the best of our knowledge, [20, 21] are the first to examine the advantages of solving the TTVSP. Specifically, [20] utilizes an iterative feedback-loop between the TT and VSP and reallocates service trips in the timetable to reduce the fleet size, while [21] propose a genetic algorithm to jointly optimize the TTVSP. In contrast to iteratively solving the TT and VSP, this work focuses on the integrated simultaneous optimization of the TTVSP. Hereinafter, regarding any references to integration, the TTVSP and TTVSP-HC will always relate to an integrated simultaneous optimization. As described in detail by [22], an integrated optimization approach enables the ability to increase efficiency and find the best overall solution that can lead to considerable cost reductions, while keeping the service quality equally good. An overview of challenges and innovations in public bus and railway transport planning is given by [23] and includes a recent review of TTVSP publications. A number of promising approaches, such as [24, 25], can be applied to solve real-world public transport planning problems. Compared to a sequential approach, solving the TTVSP leads to a significantly increased solution space and complexity. Given the nature of both planning steps, the TTVSP aims to satisfy contrary objectives. On the one hand, the timetables' quality should be maximized from the passengers' point of view, and on the other hand, the operating cost from the providers' perspective should be minimized. As described in detail by [26], there are multiple approaches to consider the two contrasting objectives: Shifting, Weighting, Pareto-front, Bi-level Programming, and Reordering. However, most approaches focus on adjusting an existing timetable to reduce the vehicle fixed and operational costs and cannot be utilized to schedule both service trips and vehicles,

and hence, are not applicable for solving the TTVSP. This study focuses on the few approaches that do not require an initial timetable for solving the TTVSP and allow flexible, non-cyclic scheduling of trips with variable headways.

To the best of our knowledge, [27] first proposed a Bi-Level Nesting Tabu Search algorithm that considers two models for solving the TTVSP heuristically without a given timetable. The upper-level model aims to minimize the total number of vehicles required and the operational costs for deadheading and idle times. In the lower-level model, a timetable with fixed headways for each route is computed, aiming to minimize the total transfer time of passengers. The bi-level integrated optimization approach considers one vehicle type and multiple depots and is applied to eight bus routes with three connecting stops. A cyclic timetable with fixed headways and vehicle schedules with increased solution quality compared to a sequential approach is computed in a short computational time.

An integrated ϵ-method to jointly solve the TTVSP is proposed in [28] and formulates the problem as a bi-objective optimization determining exact Pareto optimal fronts. The authors considered one vehicle type and up to five depots to construct timetables, as well as vehicle schedules for up to 50 lines and multiple transfer nodes. Even though the quantity of departures for each line is fixed in advance, the specific allocation of the planned trips is not predetermined, and departures between stops can vary within a time window to enable good transfer times between lines. The approach is able to solve the TTVSP in reasonable computation time for a real-world problem and is utilized to measure the compromise between the level of service and fleet cost.

A sophisticated bi-objective, bi-level integer programming model is applied by [29, 30] to elaborate on the passengers' behavior considering changes in the timetable and vehicle schedule. The upper-level model aims to minimize total operating costs related to fleet size and total passenger travel time. The lower level represents the passengers' route choice behavior regarding the upper-level results. They compute Pareto-efficient solutions for a network with up to four transit routes and four stops in reasonable computational time.

Deviating from the bi-level approach, [26] propose a diving-type matheuristic to solve large-scale multi-commodity-flow-type TTVSP models. For a real-world example of 12 disjoint bus lines, a timetable, and vehicle schedule are computed and compared to manually constructed solutions by experts and general-purpose solvers. Their approach is capable of aiding experienced planners in either obtaining better solutions or obtaining solutions more efficiently, that is, faster and with less effort.

Overall, the publications proposing an approach to solve the TTVSP report an increased solution quality. Due to the complexity of the TTVSP, most approaches are limited in their ability to fully account for the complexities of real-world public transport planning. To the best of our knowledge, an integrated solution covering a large PTN considering the interlining of vehicles, multiple vehicle types, or multiple depots that additionally aims to maximize regular clock-faced headways and connections has not yet been proposed.

2.2 Contributions

In this study, we introduce the integrated timetabling and vehicle scheduling problem with good headways and maximizing connections (TTVSP-HC) and assess the limits of exact sequential and integrated solution approaches. We demonstrate that a heuristical approach is required to solve the TTVSP-HC for large real-world instances and apply a possible heuristical scheme. Exceeding the limits of exact approaches, we propose an adaptive modular evolutionary extendable scheme (AMEES) for the TTVSP-HC capable of solving large real-world-inspired instances considering multiple connections, vehicle types, depots, and interlining of vehicles. MIP formulations for variants of the TTVSP are proposed. For the basic TTVSP case, the benefits of an integrated exact and heuristic approach compared to a sequential approach are analyzed. In addition, we provide a detailed evaluation of the advantages by considering additional quality criteria, in particular, regular 'clock-faced' headways and maximization of the number of connections. Finally, we evaluate the applicability of the proposed heuristic scheme to real-world problems of different sizes and complexity.

3 Problem Definition and the Mathematical Model

In Section 3.1, we define the classical tasks of timetabling and vehicle scheduling in public bus transportation, elaborate on the requirements for an integrated approach, and define the TTVSP and TTVSP-HC, as well as a simplified variant TTVSP-HC′. In Section 3.2, a MIP model for an exact approach for solving the TTVSP that can be incorporated in standard optimization solvers, such as *Gurobi* or *Cplex*, is presented. In Section 3.3, the basic TTVSP MIP is extended to reward clock-faced headways and timely connections.

3.1 Definition of the TT, VSP, and TTVSP(-HC)

Timetabling Problem (TT) Consecutive stops that must be covered by the same vehicle are called route. A bus line is a designated route that a bus follows to transport passengers from one location to another. Assigning departure and arrival times to each stop of a line results in a scheduled service trip. The entirety of scheduled service trips defines the timetable. Given a PTN and predefined lines covering different routes, solving the TT aims at constructing an optimal seasonal, weekday-related timetable. As briefly described in Section 2, the definition of a timetables' optimality varies greatly but usually focuses on service quality and, hence, on some aspect of passenger satisfaction. In this study, we define multiple requirements for a timetable, and we assume the desired service quality reached and passengers to be satisfied if each requirement is met. A timetable must meet the following service requirements:

Service times: The scheduled trips of each line must cover at least a defined mandatory service time. The mandatory service time is defined between a *service time start* and *service time end.*

Passengers per hour: The demand of passengers to be transported is defined at each virtual demand checkpoint. The entirety of scheduled trips must enable the buses to transport at least this passenger demand specified for each hour of service time.
Minimal headway: Minimal required time interval between two consecutive bus departures (regardless of the line) defined at each virtual demand checkpoint for predefined time periods.
Maximal headway: Maximal allowed time interval between two consecutive bus departures (regardless of the line) defined at each virtual demand checkpoint for predefined time periods.

To obtain a feasible timetable, each of these constraints must be met. However, by considering additional quality criteria, passenger satisfaction can be further increased. In this study, additional quality criteria comprise the maximization of regular "clock-faced" headways and connections between intersecting lines. Including either of these additional quality criteria significantly increases the TT complexity and classifies it as NP-hard [2, 4].

Vehicle Scheduling Problem (VSP) The timetable serves as the primary input for solving the VSP. Each scheduled trip must be assigned to a vehicle. In order to serve a trip, a vehicle must first drive to the start of a scheduled trip. This departure trip from the depot is called a pull-out, and the arrival trip back at the depot is called a pull-in trip. Trips without passengers are called deadhead trips. They connect subsequent service trips at different stops. The entire daily sequence of a vehicle's pull-out, scheduled, deadhead, and pull-in trips is called a vehicle rotation. Each vehicle rotation is executed by a different vehicle. The VSP aims to minimize the total costs comprising vehicle fixed and operational costs for serving the timetable. If fixed costs exceed the operational costs significantly, it is ensured that the number of vehicle rotations and, therefore, the number of vehicles are always minimized first, followed by the minimization of the required operational costs for deadhead and idle times required to serve each service trip. Considering multiple depots and/or multiple vehicle types classifies the VSP as NP-hard [18].

Integrated Timetabling and Vehicle Scheduling Problem (TTVSP) Solving the TTVSP aims at optimizing the timetable and corresponding vehicle schedule simultaneously to find the best overall solution. In particular, the timetable leading to the lowest-cost vehicle schedule can be computed. The objective is to minimize the overall costs while fulfilling each service quality requirement.

Integrated Timetabling and Vehicle Scheduling Problem Considering Good Headways and Maximizing Connections (TTVSP-HC) The TTVSP-HC extends the basic TTVSP by considering good headways and maximizing connections as additional quality criteria. Headways are considered good if they are "clock-faced" and regular. *Clock-faced* headways are defined by a factor of 60 (e.g., 6, 10, 12, 15, 20, and 30 min). Regularity, however, can be interpreted in various ways. In the best case, the headways of departures at each stop are equal throughout the day. With varying headway and passenger

requirements during the day, this cannot be achieved in most cases. We consider a headway regular if its duration equals both the prior and succeeding headway. Further approaches, such as minimizing the total variance of headways for each line or maximizing the total amount of equal headways for a predefined period, are possible. Intersecting lines enable connections, and a suitable time gap between arriving and departing intersecting lines ensures efficient passenger transfers. The timetable computed by solving the TTVSP-HC additionally aims to maximize the overall number of connections. Since already considering clock-faced headways in an exact approach leads to significantly increased computational time, we introduce the TTVSP-HC′ that omits regularity as a quality criterion for good headways.

When solving the TTVSP and TTVSP-HC, the following basic assumptions are made:

- The travel time between each consecutive stop of a route can vary during the day but is known in advance.
- Different vehicle types need the same time to travel.
- Departure and arrival times are measured in discrete time intervals; the smallest interval is 1 min.
- Deadheading between each depot, start, and end of a route is possible.
- The deadheading times can vary during the day but are known in advance.
- Deadheading between the start and end of a route is always faster than a scheduled trip would be.
- Both the fleet size of each vehicle type at each depot as well as parking space are considered unlimited.
- If the vehicle of a connecting line arrives within a reasonable, predefined time interval, the service quality is increased and not influenced by the passenger waiting time for the connection.

3.2 A MIP Formulation for the TTVSP

In what follows, a mathematical model for solving the basic TTVSP is proposed. The model considers service requirements, and solving it results in a cost-minimal combination of a timetable and vehicle schedule. The notation for elements of the TT and VSP is introduced first, followed by the description of the underlying network, and finally, an MIP formulation for solving the TTVSP is proposed. The basic TTVSP case is extended in Section 3.3 to reward clock-faced headways and timely connections. Table 9 in Appendix 1 summarizes the sets, parameters, functions, and variables used in the mathematical model.

Let $O = \{1, ..., m\}$ be the set of stops of the public transport network. In addition, let $L = \{1, ..., n\}$ be the set of lines. Each line $l \in L$ covers a unique ordered subset of stops $o \in O$. This ordered subset is called a route. Let R be the set of all routes and $U = \{-1, 1\}$ be the set of directions, where -1 denotes the downstream direction of a route and 1 is the upstream direction. Directions U of a route cover the same sequence of stops in reverse order. Moreover, let $R(l, u)$ be a function that returns the route r_l^u for each line $l \in L$ and direction $u \in U$. The first stop of a route is represented by $\underline{r}_l^u$, and the last stop by $\overline{r}_l^u$. Furthermore, let the function $TT(r_l^u, o_1, o_2)$ return the total travel

time in minutes between any two stops $o_1, o_2 \in r_l^u$. The service time is defined for each line and specifies the time from start to end within which service trips should be covered. For each line $l \in L$, let the set T_l define each minute of service time from start to end. For example, if Line 1 covers a service time from 6 a.m. to 7 p.m., the service time starts at minute 360 and ends at minute 1440, i.e., $T_1 = \{360, 361, ..., 1440\}$. A service trip is a route r_l^u scheduled at a specific time with the first stop $\underline{r}_l^u$ starting at any $t_l \in T_l$. For each line $l \in L$, direction $u \in U$, route r_l^u and possible starting time $t_l \in T_l$, the set S contains all possible service trips. Service trips can be carried out by different types of vehicles. Let the set V represent each possible vehicle type and depot combination, hereinafter referred to as vehicle types. We define parameter cap_v as the passenger capacity of each vehicle type $v \in V$. In addition, let set S^v contain each service trip $s \in S$ that can be carried out by a vehicle of type $v \in V$. As stated in Section 3.1, service trips are complemented by pull-out trips, pull-in trips, deadhead trips, and idle times to compose a vehicle rotation. The sets D^v, I^v, P^v_{out}, and P^v_{in} contain every possible deadhead trip, idle time, pull-out, and pull-in trip that can be carried out by each vehicle of type $v \in V$, respectively.

In the following, we will describe the network model that serves as the mathematical model's foundation. The mathematical model is based on a time-space network (TSN) formulation according to [31]. The TSN consists of nodes N and arcs A. Each node $n \in N$ represents a tuple of place and time in the network. An arc $(n_1, n_2) \in A$ connects two nodes $n_1, n_2 \in N$. Separate network layers are constructed for each vehicle type $v \in V$. For each of these vehicle type-dependent layers, let the set N^v contain every node $n \in N$, and the set A^v every arc $a \in A$ associated with an activity of vehicle type $v \in V$. Set N^v comprises nodes representing the start and the end of each possible service trip $s^v \in S^v$, as well as depot nodes representing the start of any pull-out trip $p^v_{out} \in P^v_{out}$ and end of any pull-in trip $p^v_{in} \in P^v_{in}$. Vehicle type-dependent arcs are created for each service trip $s^v \in S^v$, deadhead trip $d^v \in D^v$, pull-out trip $p^v_{out} \in P^v_{out}$, pull-in trip $p^v_{in} \in P^v_{in}$, and idle time $i_v \in I^v$. Sets A^v_s, A^v_d, A^v_i, $A^v_{p^{out}}$, and $A^v_{p^{in}}$ contain the corresponding service trip arcs, deadhead arcs, idle time arcs, pull-out arcs, and pull-in arcs depending on the vehicle type $v \in V$, respectively.

The mathematical model is formulated as a MIP and consists of binary and integer variables. For service trip, pull-out, and pull-in arcs, we define binary variables π_{a^v} that equal one if the flow on the associated arc $a^v \in A^v_s \cup A^v_{p^{out}} \cup A^v_{p^{in}}$ is one, and zero otherwise. As described in [31], the flow on deadhead and idle time arcs can exceed one. Integer decision variables σ_{a^v} set the flow for each deadhead arc $a^v \in A^v_d$ and idle time arc $a^v \in A^v_i$, respectively. Similar to [31], we extend the set of arcs A^v with one circulation flow arc a^v_c connecting the last possible pull-in with the first possible pull-out in each vehicle type-dependent layer of the TSN. The set of circulation flow arcs A^v_c is required to model a feasible flow in the TSN and to measure the total fleet size and composition. We define an integer variable θ_{a^v} that equals the flow on the corresponding arc $a^v_c \in A^v_c$, representing the number of required vehicles of type $v \in V$. Finally, let the function $cost(arc)$ define instance-specific costs for each vehicle activity represented by the corresponding arc. The costs depend on the arc's duration, traveled distance, and associated vehicle type. For circulation flow arcs A^v_c, the cost function assigns fixed costs for deploying a new vehicle of type $v \in V$.

The objective function minimizes the total operational costs of all planned service, deadhead, idle time, pull-out, and pull-in arcs, as well as the total vehicle fixed costs:

$$min \sum_{v \in V} \Big(\sum_{a^v \in A_s^v} cost(a^v)\pi_{a^v} + \sum_{a^v \in A_d^v \cup A_i^v} cost(a^v)\sigma_{a^v} + \sum_{a^v \in A_{pout}^v \cup A_{pin}^v} cost(a^v)\pi_{a^v} + \sum_{a^v \in A_c^v} cost(a^v)\theta_{a^v} \Big) \tag{1}$$

The final timetable is composed of all scheduled service trip arcs $a^v \in A_s^v$ with their corresponding variables π_{a^v} equaling one. Since the cost function $cost(arc)$ encompasses operational vehicle costs resulting from traveled distance, traveled time, and idle times for each arc, the timetable is always considered in combination with the vehicle schedule, and a simultaneous optimization of the TTVSP is achieved.

To model service requirements, we introduce virtual demand checkpoints, hereafter referred to as checkpoints. These checkpoints are positioned between two consecutive stops $o_1, o_2 \in O$ in a PTN (green symbols in Fig. 1) and define the minimal headway, the maximal headway, and the minimal amount of passengers to be transported within a defined time horizon. Let C represent the set of all checkpoints. The tuple (o_1, o_2) of two consecutive stops $o_1, o_2 \in O$ represents the position of each $c \in C$ and is denoted by co_c. For each $c \in C$, the set CT_c defines the monitored time horizon as timestamps in minutes. The first timestamp of a time horizon CT_c is denoted by $\underline{ct}_c$ and the last by $\overline{ct}_c$. For each $c \in C$, let the parameter $\underline{p}_c$ define the minimal passenger demand within the total considered time horizon CT_c. Additionally, we define the parameter $\underline{h}_c$ to be the minimal headway, and $\overline{h}_c$ to be the maximal headway at each $c \in C$. For example, if the first checkpoint is positioned between Stops 1 and 2, service requirements are defined in the time horizon between 6 and 7 a.m., at least 250 passengers have to be transported, and headways should be between a minimal headway of 3 min and maximal headway of 20 min, we define $co_1 = (1, 2)$, $CT_1 = \{360, 361, ..., 419\}$, $\underline{ct}_1 = 360$, $\overline{ct}_1 = 419$, $\underline{p}_1 = 250$, $\underline{h}_1 = 3$, and $\overline{h}_1 = 20$. Note that multiple checkpoints can define varying service requirements at the same position for different time horizons. Thus, varying minimal headways, maximal headways, and passenger demands can be specified for the same day between the same two stops.

Every service trip passing a checkpoint in the monitored time horizon CT_c has to be considered when measuring the demand fulfillment of transported passengers. As previously described, each trip arc $a_s^v \in A_s^v$ refers to a route $r_l^u \in R_l^u$ with $\underline{r}_l^u$ departing at a specific timestamp in the service time $t_l \in T_l$. For each $c \in C$, the set AP_c^v contains service trip arcs $a_s^v \in A_s^v$ of each vehicle type that meets the following conditions: first, the stops co_c are a subset of route r_l^u associated with the service trip a_s^v; second, the specific start time t_l plus the travel time from the first stop $\underline{r}_l^u$ of the route to the first stop $o_1 : (o_1, o_2) \in co_c$ is within the monitored time horizon. Therefore, $\underline{ct}_c \leq t_l + TT(r_l^u, \underline{r}_l^u, o_1 : (o_1, o_2) \in co_c) \leq \overline{ct}_c$.

The minimal and maximal headway defined by a checkpoint ensures that the number of service trips, and hence the departure of vehicles, planned within time horizon CT_c is neither too frequent nor too rare. A minimal headway (e.g., 3 min) is violated

if more than one service trip is scheduled in this time interval. Opposed to the set AP_c^v that contains each service trip arc that passes the checkpoint in the monitored time horizon, we define a set $\underline{AH}_c^v$ for each vehicle type $v \in V$ and checkpoint $c \in C$ that contains multiple subsets $\underline{AH}_{c,ct_c}^v$ for each minute in the time horizon $ct_c \in CT_c$. In particular, $\underline{AH}_c^v = \{\underline{AH}_{c,\underline{ct}_c}^v, \underline{AH}_{c,\underline{ct}_c+1}^v, ..., \underline{AH}_{c,\overline{ct}_c-\underline{h}_c+1}^v\}$. For each checkpoint $c \in C$, vehicle type $v \in V$, and minute $ct_c \in CT_c$, the set $\underline{AH}_{c,ct_c}^v$ contains service trip arcs $a_s^v \in A_s^v$ that meet the following conditions: first, the stops co_c are a subset of route r_l^u associated with the service trip a_s^v; second, the specific start time t_l plus the travel time from the first stop $\underline{r}_l^u$ of the route to the first stop $o_1 : (o_1, o_2) \in co_c$ is between ct_c and ct_c plus the minimal headway $\underline{h}_c$. Therefore, $ct_c \leq t_l + TT(r_l^u, \underline{r}_l^u, o_1 : (o_1, o_2) \in co_c) \leq ct_c + \underline{h}_c$. For example, if the minimal headway is 3 min between Stops 5 and 6, between 6 a.m. and 7 a.m., that means $co_1 = (5, 6)$, $CT_1 = \{360, 361, ..., 419\}$, and $\underline{h}_c = 3$. For each considered vehicle type and line passing the checkpoint, each set $\underline{AH}_{1,ct_1}^v$ contains three trip arcs a_s^v connecting two nodes $n_1, n_2 \in N$ for each minute in $ct_1 \in CT_1$:

Example 1

$$
\begin{aligned}
\underline{AH}_{1,360}^v = \{&a_{s1}^v = ((\underline{r}_l^u, 360 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 360 + TT(r_l^u, 6, \overline{r}_l^u))),\\
&a_{s2}^v = ((\underline{r}_l^u, 361 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 361 + TT(r_l^u, 6, \overline{r}_l^u))),\\
&a_{s3}^v = ((\underline{r}_l^u, 362 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 362 + TT(r_l^u, 6, \overline{r}_l^u)))\},\\
\underline{AH}_{1,361}^v = \{&a_{s2}^v = ((\underline{r}_l^u, 361 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 361 + TT(r_l^u, 6, \overline{r}_l^u)))),\\
&a_{s3}^v = ((\underline{r}_l^u, 362 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 362 + TT(r_l^u, 6, \overline{r}_l^u))),\\
&a_{s4}^v = ((\underline{r}_l^u, 363 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 363 + TT(r_l^u, 6, \overline{r}_l^u)))\},\\
&...,\\
\underline{AH}_{1,417}^v = \{&a_{s58}^v = ((\underline{r}_l^u, 417 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 417 + TT(r_l^u, 6, \overline{r}_l^u))),\\
&a_{s59}^v = ((\underline{r}_l^u, 418 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 418 + TT(r_l^u, 6, \overline{r}_l^u))),\\
&a_{s60}^v = ((\underline{r}_l^u, 419 - TT(r_l^u, \underline{r}_l^u, 5)), (\overline{r}_l^u, 419 + TT(r_l^u, 6, \overline{r}_l^u)))\}
\end{aligned}
$$

Figure 2 illustrates the structure of Example 1. Black horizontal lines represent the dimensions of Stops 5 and 6 in $co_1 = (5, 6)$. The timestamp of each arc passing Stop 5 is displayed above the horizontal line of Stop 5. Arc a_{s1}^v, for example, passes Stop 5 at timestamp 360 and Stop 6 at timestamp 360 + TT(r_l^u, 5, 6). Dotted arrows illustrate the arrival of an arc at Stop 5 and a small plateau illustrates the time interval between arrival and departure at Stop 5. The demand checkpoint c_1, displayed as a horizontal blue line, sets requirements to arcs passing Stops 5 and 6 and is positioned right after Stop 5. Since the minimal headway is 3 min, each set $\underline{AH}_{c,ct_c}^v$ comprises three arcs (indicated by colored brackets). The set $\underline{AH}_{1,360}^v$, for example, comprises the first three arcs passing Stop 5: a_{s1}^v, a_{s2}^v, and a_{s3}^v (see the green bracket). As described in Example 1, the same arc can be included in multiple sets of $\underline{AH}_c^v$. Arcs a_{s2}^v and a_{s3}^v are, for example, included in both $\underline{AH}_{1,360}^v$ and $\underline{AH}_{1,361}^v$. Arc a_{s3}^v is additionally included in a third set $\underline{AH}_{1,362}^v$. Since only discretized minutes are considered when planning service trips, the amount of trips in each $\underline{AH}_{c,ct_c}^v$ for each $ct_c \in CT_c$ equals

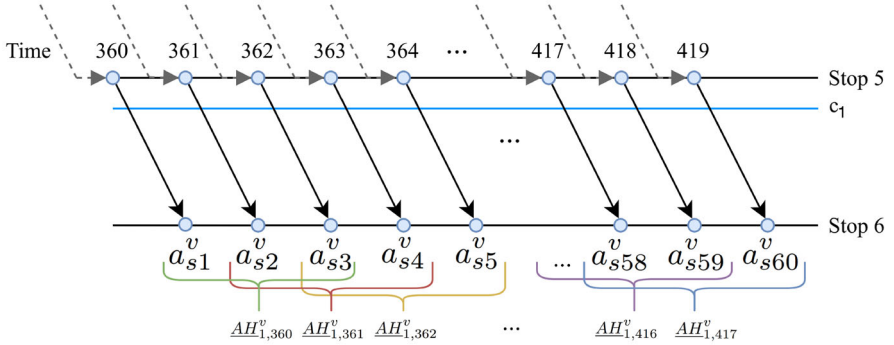


Fig. 2 Illustration of Example 1

the minimal headway $\underline{h}_c$. However, if lines run parallel in the PTN and pass the same checkpoint, the number of trips in each $\underline{AH}^v_{c,ct_c}$ is multiplied by the total number of lines passing the checkpoint.

The maximal headway is violated if the interval between two consecutive service trips is too long, hence leading vehicles to depart too rarely. Similarly to $\underline{AH}^v_c$, for each vehicle type $v \in V$ and checkpoint $c \in C$, the set $\overline{AH}^v_c$ contains multiple subsets $\overline{AH}^v_{c,ct_c}$ for each minute in the time horizon $ct_c \in CT_c$. In particular, $\overline{AH}^v_c = \{\overline{AH}^v_{c,\underline{ct}_c}, \overline{AH}^v_{c,\underline{ct}_c+1}, ..., \overline{AH}^v_{c,\overline{ct}_c-\overline{h}_c+1}\}$. The set $\overline{AH}^v_{c,ct_c}$ contains every service trip $a^v_s \in A^v_s$ within the maximal headway $\overline{h}_c$, therefore, $ct_c \leq t_l + TT(r^u_l, \underline{r}^u_l, o_1 : (o_1, o_2) \in co_c) \leq ct_c + \overline{h}_c$. For example, if the maximal headway is 20 min between Stops 5 and 6, between 6 and 7 a.m., that means $co_1 = (5, 6)$, $CT_1 = \{360, 361, ..., 419\}$, and $\overline{h}_c = 20$. For each considered vehicle type and line passing the checkpoint, each set $\overline{AH}^v_{1,ct_1}$ contains 20 trip arcs a^v_s connecting two nodes $n_1, n_2 \in N$ for each minute in $ct_1 \in CT_1$:

Example 2

$$
\begin{aligned}
\overline{AH}^v_{1,360} = \{&a^v_{s1} = ((\underline{r}^u_l, 360 - TT(r^u_l, \underline{r}^u_l, 5)), (\overline{r}^u_l, 360 + TT(r^u_l, 6, \overline{r}^u_l))),\\
&a^v_{s2} = ((\underline{r}^u_l, 361 - TT(r^u_l, \underline{r}^u_l, 5)), (\overline{r}^u_l, 361 + TT(r^u_l, 6, \overline{r}^u_l))),\\
&...,\\
&a^v_{s20} = ((\underline{r}^u_l, 379 - TT(r^u_l, \underline{r}^u_l, 5)), (\overline{r}^u_l, 379 + TT(r^u_l, 6, \overline{r}^u_l)))\},\\
\overline{AH}^v_{1,361} = \{&a^v_{s2} = ((\underline{r}^u_l, 361 - TT(r^u_l, \underline{r}^u_l, 5)), (\overline{r}^u_l, 361 + TT(r^u_l, 6, \overline{r}^u_l))),\\
&a^v_{s3} = ((\underline{r}^u_l, 362 - TT(r^u_l, \underline{r}^u_l, 5)), (\overline{r}^u_l, 362 + TT(r^u_l, 6, \overline{r}^u_l))),\\
&...\\
&a^v_{s21} = ((\underline{r}^u_l, 380 - TT(r^u_l, \underline{r}^u_l, 5)), (\overline{r}^u_l, 380 + TT(r^u_l, 6, \overline{r}^u_l)))\},\\
&...,
\end{aligned}
$$

$$\begin{aligned}\overline{AH}^{v}_{1,400} = \{&a^{v}_{s41} = ((\underline{r}^{u}_{l}, 400 - TT(r^{u}_{l}, \underline{r}^{u}_{l}, 5)), (\overline{r}^{u}_{l}, 400 + TT(r^{u}_{l}, 6, \overline{r}^{u}_{l}))),\\ &a^{v}_{s42} = ((\underline{r}^{u}_{l}, 401 - TT(r^{u}_{l}, \underline{r}^{u}_{l}, 5)), (\overline{r}^{u}_{l}, 401 + TT(r^{u}_{l}, 6, \overline{r}^{u}_{l}))),\\ &\ldots\\ &a^{v}_{s60} = ((\underline{r}^{u}_{l}, 419 - TT(r^{u}_{l}, \underline{r}^{u}_{l}, 5)), (\overline{r}^{u}_{l}, 419 + TT(r^{u}_{l}, 6, \overline{r}^{u}_{l})))\}\end{aligned}$$

Similar to the set $\underline{AH}^{v}_{c,ct_c}$, the number of service trip arcs in $\overline{AH}^{v}_{c,ct_c}$ equals the maximal headway $\overline{h}_c$ for each $ct_c \in CT_c$. If lines run parallel in the PTN and pass the same checkpoint, the number of trips in each $\overline{AH}^{v}_{c,ct_c}$ is multiplied by the total number of lines passing the checkpoint.

The constraints of the model to meet all service requirements are formulated as follows:

$$\sum_{v\in V}\sum_{a^v\in\underline{AH}^{v}_{c,ct_c}} \pi_{a^v} \le 1 \qquad \forall c \in C, ct_c \in CT_c \tag{2}$$

$$\sum_{v\in V}\sum_{a^v\in\overline{AH}^{v}_{c,ct_c}} \pi_{a^v} \ge 1 \qquad \forall c \in C, ct_c \in CT_c \tag{3}$$

$$\sum_{v\in V}\sum_{a^v\in AP^{v}_{c}} cap_v \pi_{a^v} \ge \underline{p}_c \qquad \forall c \in C \tag{4}$$

$$\begin{aligned}&\sum_{(i,n)\in A^{v}_{s}\cup A^{v}_{p^{out}}\cup A^{v}_{p^{in}}} \pi_{(i,n)} + \sum_{(i,n)\in A^{v}_{d}\cup A^{v}_{i}} \sigma_{(i,n)} + \sum_{(i,n)\in A^{v}_{c}} \theta_{(i,n)}\\ &- \sum_{(n,j)\in A^{v}_{s}\cup A^{v}_{p^{out}}\cup A^{v}_{p^{in}}} \pi_{(n,j)} - \sum_{(n,j)\in A^{v}_{d}\cup A^{v}_{i}} \sigma_{(n,j)}\\ &- \sum_{(n,j)\in A^{v}_{c}} \theta_{(n,j)} = 0 \quad \forall v \in V, n \in N^v\end{aligned} \tag{5}$$

$$\pi_{a^v} \in \{0,1\}\ \forall v \in V, a^v \in A^{v}_{s} \cup A^{v}_{p^{out}} \cup A^{v}_{p^{in}} \tag{6}$$

$$\sigma_{a^v} \in \mathbb{N}\ \forall v \in V, a^v \in A^{v}_{d} \cup A^{v}_{i} \tag{7}$$

$$\theta_{a^v} \in \mathbb{N}\ \forall v \in V, a^v \in A^{v}_{c} \tag{8}$$

For each checkpoint, constraints (2) guarantee that the minimal headway is always maintained. No two service trips from any set of service trips $\underline{AH}^{v}_{c,ct_c}$ can be scheduled, and vehicles are prevented from departing within too small a time interval. As illustrated in Example 1, trip arc $a^{v}_{s_3}$ is part of $\underline{AH}^{v}_{1,360}$, $\underline{AH}^{v}_{1,361}$, and inherently of $\underline{AH}^{v}_{1,362}$. Consequently, by scheduling $a^{v}_{s_3}$, constraints (2) prohibit the scheduling of any other trip arc in these three sets, ensuring the minimal headway requirement is always met. On the other hand, similarly structured constraints (3) ensure for each checkpoint that the departure of two consecutive trips never exceeds the maximal

headway. As illustrated in Example 2, each set $\overline{AH}^{v}_{c,ct_c}$ contains trips within the range of the maximal headway. By requiring the planning of at least one service trip from each set of service trips $\overline{AH}^{v}_{c,ct_c}$, it is ensured that the maximal headway is never exceeded. Constraints (4) guarantee that at least the minimal demand of passengers can be transported. Lastly, the flow-conservation constraints (5) ensure that every flow unit representing a vehicle entering a node also leaves it.

3.3 A Mathematical Model for Solving the TTVSP-HC

A valid solution to the proposed TTVSP formulation satisfies the defined service requirements for the timetable while minimizing the operational costs for vehicles. The model (1)–(8) can be extended to consider good headways and reward timely connections and thus solve the TTVSP-HC. However, preliminary experiments have shown that these extensions increase the model's complexity substantially. As a result, we introduce the *TTVSP-HC′*, that omits the regularity of headways. Thus, headways in the TTVSP-HC′ are already considered good if they are clock-faced.

To reward timely connections, we define the set X to contain every two routes r^u_l that intersect, and for such an intersection, a well-scheduled passenger transfer is desired by the public transport planner. For each connection $x \in X$, $\underline{w}_x$ defines the time in minutes it takes for a passenger to reach the intersecting route. The upper waiting time limit for a bus serving the connecting route to arrive is defined by parameter $\overline{w}_x$ for each $x \in X$. For example, connections of the upstream direction of Line 1 to the downstream direction of Line 3 are desired. It takes 2 min for a passenger to walk from Line 1 to Line 3, and connections should only be rewarded if passengers wait a maximum of 5 min. Then, $x = (r^1_1, r^{-1}_3)$, $\underline{w}_x = 2$, and $\overline{w}_x = 5$. Additionally, for each $x \in X$, let the set AC_x contain every tuple of service trip arcs $(a_1, a_2) : a_1, a_2 \in A^v_s$ that meet the following conditions: first, tuple x contains the routes associated with both a_1 and a_2; second, at their intersecting stop, the time interval between the arrival of a_1 and departure of a_2 is between $\underline{w}_x$ and $\overline{w}_x + \underline{w}_x$. We define a binary variable $b^x_{a_1,a_2}$ for each intersection $x \in X$ and each tuple of trip arcs $(a_1, a_2) \in AC_x$ that is one if both service trips are scheduled and zero otherwise.

To reward clock-faced headways, the set H^{clock} contains every time interval that is a factor of 60 and should be rewarded, that is, $H^{clock} = \{5, 6, 10, 12, 15, 20, 30\}$. Since passengers usually do not have to plan their journey for repeating short headways, headways of less than 5 min, in particular, are not rewarded. The set HO contains the stops $o \in O$ where clock-faced headways are rewarded. For a PTN with no parallel lines, HO will likely contain the start of each route $r^u_l : r^u_l \in R^u_l$. If lines run parallel, stops covered by multiple lines can be included in HO to additionally reward clock-faced headways from departures of different lines. For each stop $o \in HO$, the set AH_o contains every tuple of service trip arcs $a_1, a_2 \in A^v_s$ that depart in a clock-faced interval. Importantly, clock-faced headways are only rewarded if there is no service trip planned in between two departures $ah_o \in AH_o$. Hence, we define a function $HC(o, a_1, a_2)$ that returns every service trip arc $a^v_s \in A^v_s$ that can possibly be scheduled between two service trip arcs $(a_1, a_2) : a_1, a_2 \in A^v_s$ at stop $o \in HO$. Further, we introduce an integer constant M that equals the highest value that the

function $HC(o, a_1, a_2)$ can possibly return. We define a binary variable $b^o_{a_1,a_2}$ for each stop $o \in O$ and each tuple of trip arcs $(a_1, a_2) \in AH_o$ that is one if both service trips are scheduled and no trip arc $a \in HC(o, a_1, a_2)$ is planned, and zero otherwise. Finally, parameters α and β can be set to weigh the impact of rewarding headways and connections. The objective function 1 is extended to reward good headways and connections:

$$\begin{aligned} min \sum_{v \in V} \sum_{a^v \in A^v_s} cost(a^v)\pi_{a^v} + \sum_{a^v \in A^v_d \cup A^v_i} cost(a^v)\sigma_{a^v} \\ + \sum_{a^v \in A^v_{pout} \cup A^v_{pin}} cost(a^v)\pi_{a^v} + \sum_{a^v \in A^v_c} cost(a^v)\theta_{a^v} \\ - \alpha \sum_{x \in X} \sum_{(a_1,a_2) \in AC_x} b^x_{a_1,a_2} - \beta \sum_{o \in HO} \sum_{(a_1,a_2) \in AH_o} b^o_{a_1,a_2} \end{aligned} \tag{9}$$

Each clock-faced headway and connection reached improves the objective value. The following constraints only allow b^x_{ac} to equal one if both trip arcs $(a_1, a_2) \in AC_x$ are planned:

$$\pi_{a_1} + \pi_{a_2} \geq 2b^x_{(a_1,a_2)} \qquad \forall x \in X, (a_1, a_2) \in AC_x \tag{10}$$

Similar to (10), the bonus for a clock-faced headway is only granted when both service trips are scheduled. An additional check to see if no other trip is planned between the two consecutive trips $(a_1, a_2) \in AH_o$ is required:

$$\pi_{a_1} + \pi_{a_2} \geq 2b^o_{(a_1,a_2)} \qquad \forall o \in HO, (a_1, a_2) \in AH_o \tag{11}$$

$$\sum_{a \in HC(o,a_1,a_2)} \pi_a \leq (1 - b^o_{(a_1,a_2)})M \quad \forall o \in HO, (a_1, a_2) \in AH_o \tag{12}$$

$$b^x_{(a_1,a_2)} \in \{0, 1\} \quad \forall x \in X, (a_1, a_2) \in AC_x \tag{13}$$

$$b^o_{(a_1,a_2)} \in \{0, 1\} \quad \forall o \in HO, (a_1, a_2) \in AH_o \tag{14}$$

We suggest the extended model (1)–(14) to solve the TTVSP-HC′ for small instances. Please note that the big-M formulation of constraint (12) especially might lead to weak LP-relaxations and hence increase computational time notably. Future research could evaluate formulations that enable stronger LP-relaxations and might increase efficiency (see Outlook in Section 6). However, it is not expected that a varying formulation is capable of computing optimal solutions for the real-world TTVSP-HC′ or TTVSP-HC in reasonable computational time. In the following section, a heuristic approach for solving the TTVSP-HC is proposed and compared with the exact MIP model in Section 5.

4 Heuristically Solving the TTVSP-HC with the AMEES

We demonstrate the possibility of heuristically solving the TTVSP-HC for large real-world inspired instances in reasonable computational time by utilizing a possible evolutionary scheme. The proposed AMEES is able to consider a variety of cost and service quality criteria in addition to vehicle fixed and operational costs. Early-stage experiments suggested that recombining promising individuals through cross-over operations leads to invalid or deteriorated solutions frequently, and such a procedure required too much effort to restore validity. To accelerate intensification, applying various mutations offered promising results. Therefore, the main novelty of the AMEES for public transport planning is the inclusion of an extendable variety of numerous Mutation Operators (MOs), which provide an overall good solution progress in the optimization process. In the following, the design of the AMEES solving the TTVSP-HC is described. First, we illustrate the AMEES's structure and its specifications for solving a TTVSP-HC. Following this, the generic solution manager applicable to different optimization problems is outlined, and finally, an overview of MOs directed at optimizing the TTVSP-HC is given.

4.1 An AMEES for TTVSP-HC

The AMEES improves iteratively a set of TTVSP-HC solutions and, as illustrated in Fig. 3, consists of five major parts: the problem-specific input (1), fitness function (2), mutation operators (3), a generic solution manager (4), and output (5).

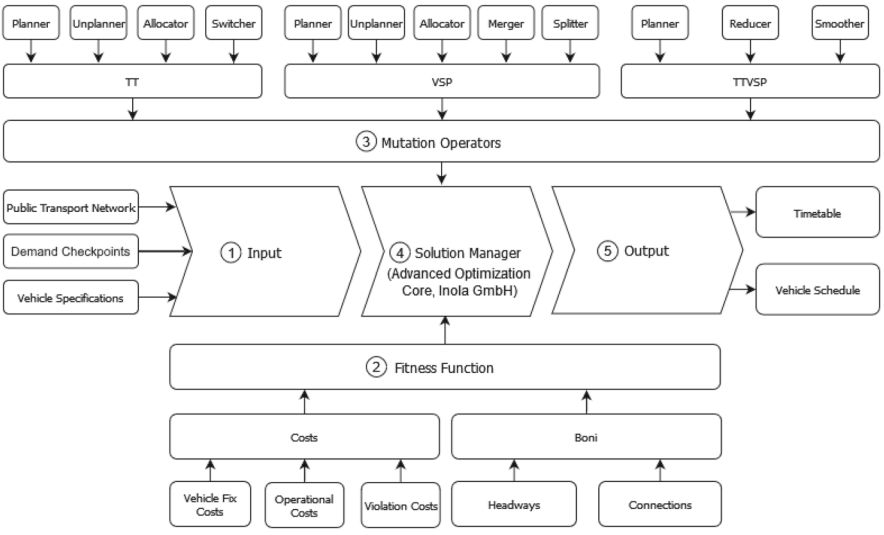


Fig. 3 Components of the AMEES

First, the problem-specific input is processed to suit an evolutionary approach. The input comprises a PTN; the positioning and requirements of checkpoints; and vehicle specifications, such as capacities and fixed and operational costs. Besides loading the input data, its structure is mapped to build individuals. When solving a TTVSP-HC, an individual represents a TTVSP-HC solution, that is, a timetable and corresponding vehicle schedule. Analogous to the exact formulation, the timetable is represented by variables $\pi_{a_s^v}$. For each $a_s^v \in A_s^v$, the variable $\pi_{a_s^v}$ equals one if the associated service trip is scheduled, and zero otherwise. The vehicle schedule is represented as a dynamic list of vehicle rotations. Each rotation is associated with one vehicle and contains a list of the covered trips $a_s^v \in A_s^v$, deadheads $a_d^v \in A_d^v$, idle times $a_i^v \in A_s^v$, pull-outs $a_{p^{out}}^v \in A_{p^{out}}^v$, pull-ins $a_{p^{in}}^v \in A_{p^{in}}^v$, and an associated depot. The AMEES can start from a diverse population and include individuals computed by previous executions of the AMEES or other construction heuristics. For the experiments in this study, the initial population comprises a number of equal individuals, each containing a timetable without a scheduled trip (every $\pi_{a_s^v}$ equals zero) and no vehicle rotations. Applying a variety of MOs to a sufficiently large initial population enables promising diversification in the solution process. However, if the solution quality converges, the population size increases dynamically to a maximum of 50,000 individuals within the runtime of the AMEES. Due to the possibility of applying MOs to a more extensive range of individuals, the likelihood of escaping local optima increases. A problem-specific fitness function is required to evaluate a solution's quality and to quantify the possible improvements after applying an MO. The fitness of a TTVSP-HC individual consists of costs and bonuses. Costs are divided into vehicle fixed and operational costs, as well as constraint violation costs. Vehicle fixed costs cover the total costs of acquiring each vehicle. Operational costs comprise the cost per distance a vehicle traveled for service, deadhead, pull-out, and pull-in trips, as well as costs per idle minute a vehicle is not in the depot. Violating requirements for minimal headways, maximal headways, service times, and passengers to be transported results in a significantly decreased fitness value. On the other hand, bonuses for good headways and connections reached increase an individual's fitness. The AMEES utilizes various enhancing MOs to improve an individual's fitness. They can be categorized by their primary objective of improving the timetable, vehicle schedule, or both, and can be further classified. A simple MO could, for example, flip a random binary value within the trip schedule, resulting in planning or unplanning a trip. The set of all MOs are:

- **Adaptively** applied within adjusting probability given their past performance.
- **Modularly** exchangeable and modifiable without being directly dependable on each other.
- **Extendable** to consider new objectives or increase the overall performance without affecting other MOs.

The entirety of MOs enables the solving of the TTVSP-HC. Whereas some MOs individually improve a solution, others might decrease an individual's fitness in the short term but enable an overall improvement. A detailed overview and categorization of MOs utilized for solving the TTVSP-HC are given in Section 4.3. The generic solution manager is applicable to different optimization problems and regulates the probability of MOs being applied in each generation. An overview of the solution

manager's operating principle is given in Section 4.2. After reaching a termination criterion, for example, total runtime or solution quality, the best individual is selected and the output, that is, the timetable and vehicle schedule is provided.

4.2 Solution Manager

The Solution Manager is a generic evolutionary algorithm applicable to numerous optimization problems. However, it can only operate embedded in the entirety of a problem-specific solution scheme. The implementation of Algorithm 1, *Advanced Optimization Core* (AOC), has been made available for this research by the planning software provider *Inola GmbH*. The AOC manages the application of MOs to individuals and each generation's population composition. Algorithm 1 illustrates the concept of applying and evaluating MOs in each generation. Initially, two conditions are checked: if the maximal number of generations is reached (Line 1) and if the current population size is smaller than the maximal size of the population (Line 2). If the first condition is reached, the AOC terminates. Other termination criteria are possible, such as reaching a specific objective value or a runtime limit. A new generation starts whenever the population has grown to its current maximum limit. If the population limit is not exceeded, each individual is iterated consecutively (Line 3). If no MO has been applied to an individual yet, a suitable MO is stochastically selected (Line 5). The likelihood of an MO being applied is determined in proportion to its past performance. Depending on the positive and negative impact of applying an MO to an individual, its likelihood of being selected in future iterations changes accordingly; the more the fitness value increases, the higher the future probability of an MO being selected is. Notably, the pre-mutated individual stays in the population but will not mutate again in the same generation. This guarantees that even if applying an MO leads to decreased fitness, the superior, initial individual can succeed to the next generation. However, the mutated individual can mutate again in the same generation if the population size limit is not reached after the first iteration. This mechanism is utilized to examine synergies between consecutively applied MOs. Consider an individual that originated from mutating in the same generation. Algorithm 1 does not only measure the performance of a single MO, but the combined performance of each MO applied to the individual (Line 9); for example, if two MOs have been applied to an individual and the fitness increased more than it would have by applying each MO individually, the likelihood of both MOs being selected in future iterations is further increased. As a result, MOs that do not lead to an improved solution individually but synergize well with other MOs are still rewarded. Finally, if the population size exceeds its limit, only the strongest individuals progress to the next generation (Line 16). Note that if there is a strict validity criterion for individuals, the AOC's primary goal is to compute feasible solutions and minimize the total costs subsequently. The solution manager is applicable to various optimization problems. However, its success highly depends on the quality of MOs. The following section describes the proposed MOs utilized in the AOC to solve the TTVSP-HC specifically.

Algorithm 1 Generic Solution Manager - Advanced Optimization Core

```
1: while generationIteration < maximalGeneration do
2:     while population.size() < maximalPopulationSize do
3:         for individual in Population do
4:             if not individual.isMutated then
5:                 MO = rollMOfromAllMOs();
6:                 newIndividual = individual.copy();
7:                 newIndividual.applyMO(MO);
8:                 newIndividual.calculateFitness();
9:                 MO.evaluateImpact();
10:                population.add(newIndividual);
11:                individual.isMutated = TRUE;
12:            end if
13:        end for
14:    end while
15:    Population.sortByFitness();
16:    Population.remove(Population.Size() - initialPopulationSize);
17:    generationIteration ++;
18: end while
```

4.3 Mutation Operators

MOs aim to modify an individual to increase their fitness. Importantly, an MO does not have to yield an improvement directly, but can also be utilized to alter a solution, enabling another MO to achieve an overall improvement. A simple MO could, for example, try to merge two separate vehicle rotations into a single one. However, to reduce the overall runtime and have the AMEES act as a reliable optimizer rather than a randomizer, MOs should aim to smartly apply small changes leading directly to an improved individual or pave the way to an improvement. When solving a large real-world TTVSP-HC, a total of 33 problem-specific MOs are utilized. Each MO fulfills a different task, and the entirety of MOs serves to solve the TTVSP-HC. MOs applied to the TTVSP-HC can be put into three categories: improving the timetable, improving the vehicle schedule, or improving both. Table 1 gives an overview of proposed MOs utilized in the AMEES and categorizes them by their *objective* (Obj.) and *Class*. A total of four classes are dedicated to improving the TT, while four classes are focused on improving the VSP, and one class is dedicated to improving both the TT and VSP simultaneously. The concept and procedural details of the proposed MOs associated with each class are described in detail in Appendices 3–11. Table 10 in Appendix 2 extends the sets, functions, and parameters of the exact model given in Appendix 1 for the detailed description of the operating principle of each MO.

MOs aiming to improve the timetable are separated into four classes: *T-Planner*, *T-Unplanner*, *T-Switcher*, and *T-Allocator*. In total, a pool of 21 timetabling MOs is rapidly applied to each individual. Several T-Planner and T-Unplanner MOs aim at planning or unplanning particular trips to improve the timetable. T-Switcher MOs specifically change the value of two trips, and T-Allocator MOs of multiple trips simultaneously. The class of T-Planner, for example, contains eight distinct MOs, each scheduling a trip while aiming to achieve a different enhancement.

Table 1 Overview of proposed MOs utilized in the AMEES

Obj.	Class	Associated MOs	Appendix
TT	T-Planner	MaxHeadwayViolated, MaxHeadway-ViolatedDiffLine, MinHeadway, PassengerMissing, Regular, Connection, Clockheadway, ServiceTime	3 (Table 11)
TT	T-Unplanner	MinHeadwayViolated, PassengerOverplanned	4 (Table 12)
TT	T-Switcher	ServiceTimeStartLatest, ServiceTime- EndEarliest, MinHeadwayViolated, MaxHeadwayViolated, ClockHeadway, Connection	5 (Table 13)
TT	T-Allocator	MaxHeadway, MinHeadway, Clockheadway, HeadwayIncreasing, Regularity	6 (Table 14)
VSP	V-Planner	Vehicle	7 (Table 15)
VSP	V-Merger	Extensive, Best	8 (Table 16)
VSP	V-Splitter	BestFit, HighCost	9 (Table 17)
VSP	V-Allocator	CostReduction, TotatTimeReduction, DeadheadReduction, IdleTimeReduction	10 (Table 18)
TT+VSP	TTVSP	EfficientVehicleRotation, VehicleReducer, VehicleTripSmoother	11 (Table 19)

The vehicle schedule is represented as a vector of vehicle rotation objects. Each rotation contains a list of the covered trips, deadhead trips, idle times, and an associated depot. In total, a pool of nine MOs aiming to improve an individual's vehicle schedule is divided into four classes: *V-Planner*, *V-Merger*, *V-Splitter*, and *V-Allocator*. New vehicle rotations are created via V-Planner to cover unassigned trips. V-Merger MOs reduce the total number of required buses by merging trips of two rotations into a single one. Long idle times or too much deadheading can lead to inconveniently planned vehicle rotations. V-Splitter MOs divide these into separate new rotations, which can be merged with other rotations in future iterations. Resulting in lower operational costs, V-Allocator MOs reallocate trips between rotations.

Even though the TT-MOs only aim to adjust the timetable, interdependencies between the TT and VSP are implicitly considered. Since the solution is not feasible if a trip is not assigned to a vehicle, the V-Planner is always applied to newly scheduled or reallocated trips that cannot be covered by the same vehicle anymore. The V-Merger attempts to unite the new vehicle with existing ones. If this is not possible, the overall costs increase and the individual's fitness decreases accordingly. This negative impact can be implicitly associated with the initially applied TT-MO and will reduce its likelihood of being applied in future iterations. Conversely, TT-MOs synergize well with the VSP-MOs, and the current vehicle schedule will be prioritized in future iterations. Note that due to the assumptions made in Section 3, a trip can always be replaced with a deadhead trip. As a result, unplanning or rescheduling a trip will always keep existing vehicle rotations feasible.

Finally, a pool of three MOs listed and described in Table 19 in Appendix 11 aims to explicitly improve both the timetable and the vehicle schedule simultaneously. The *EfficientVehicleRotationPlanner* MO adds a new vehicle with multiple trips. The newly planned trips are selected to reduce constraint violations and additionally have short

idle times and little deadheading when planned in the same vehicle rotation. Aiming to decrease the total required vehicles, the *VehicleReducer* MO selects multiple vehicle rotations and allocates their trips such that fewer vehicles are required. Additionally, trips leading to high idle times or long deadheading are unplanned. Finally, the *VehicleTripSmoother* MO reallocates trips within rotations such that the idle and deadhead times are reduced. However, the quality of the timetable must remain equally high or even be improved.

5 Experiments

In this section, we evaluate the proposed approaches on real-world-inspired instances provided by *INIT Mobility Software Solutions GmbH*, a leading worldwide public transport planning company. We assess solving the TTVSP-HC in comparison to the TTVSP and the sequential approach. In addition, we examine the limits of the exact approaches and evaluate the applicability of the heuristic AMEES. The considered real-world-inspired instances and parameter settings are described in the following section (Section 5.1). Section 5.2 evaluates the results of exactly sequentially solving the TT and VSP and integrated optimization of the TTVSP. The limitations and benefits of exactly solving the TTVSP-HC′ are examined in Section 5.3. Section 5.4 evaluates the applicability of heuristically solving the TTVSP and TTVSP-HC with the AMEES. All computational tests are carried out on an Intel(R) Core(TM) i9-10900 CPU with 2.8GHz and 32GB RAM. The AMEES is implemented in C++ v 14.31.3103, and for solving MIPs we use Gurobi v 9.1.2.

5.1 Instances and Parameter Settings

We consider six instances of different problem sizes and complexity. The instances are provided by *INIT Mobility Software Solutions GmbH* and describe differently sized real-world-inspired PTNs. Table 2 provides an overview of the main properties of the instances denoted by I1 to I6, and Fig. 4 illustrates the structure of the corresponding PTN.

Table 2 General properties of problem instances

Instance	*#Lines*	#Stops	*Total service time (minutes)*	*#Checkpoints*	#Intersections
I1	2	5	120	2	1
I2	3	13	720	6	4
I3	3	13	2880	6	4
I4	5	29	11,700	11 (54)	8
I5	5	29	58,500	11 (54)	8
I6	25	145	58,500	55 (270)	40

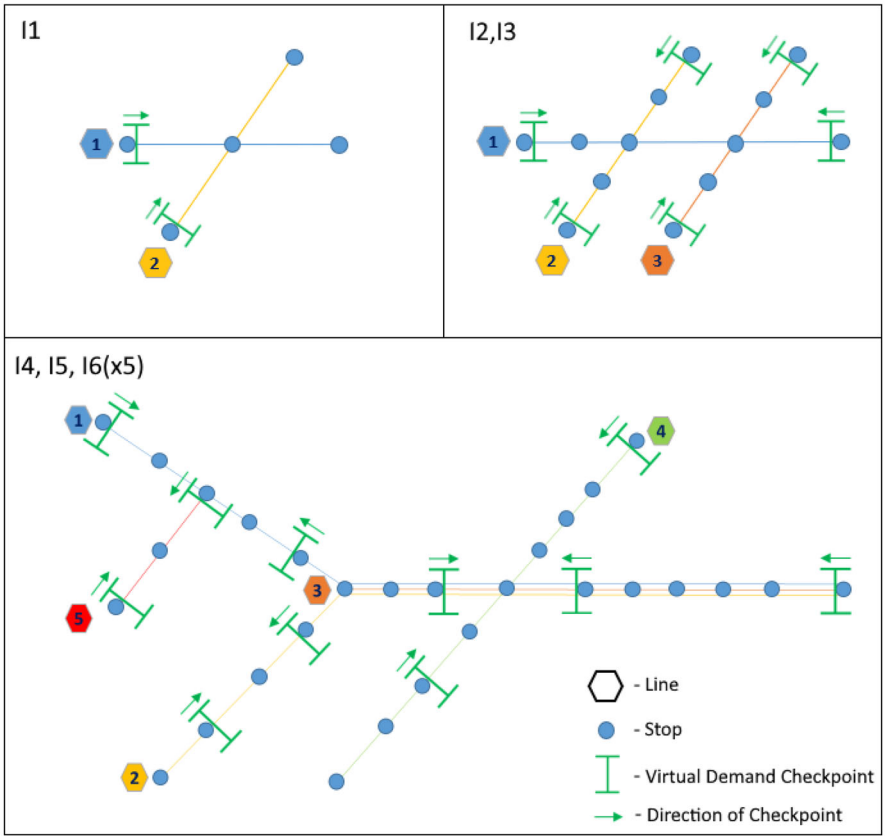


Fig. 4 Visualization of instances

In Table 2, for each *Instance*, the column *#Lines* indicates the number of lines, and the column *#Stops* the number of stops covered in the underlying PTN. The *Total Service Time* is composed of the maximum time each line is served in each direction. If a PTN would, for example, comprise three bi-directional lines with 8 h of service time each, the total service time would equal 2880 min (3 lines * 2 directions * 8 h * 60 min). The column *#Checkpoints* indicates the number of checkpoints in the PTN. For I4, I5, and I6, checkpoints set requirements for different time periods. In this case, the total considered quantity of time periods is displayed in brackets after the number of considered checkpoints. The last column *#Intersections* shows the number of lines intersecting in the PTN.

The specifications and structure of the PTN in each instance are illustrated in Fig. 4. In the PTN, each line is indicated by a different color, and their routes connect different stops (blue circle). Green symbols represent checkpoints, and arrows over the checkpoints indicate the direction of a route to which requirements are set. Instance I1 represents a PTN comprising two one-directional lines, with each line covering three

stopping points. Both lines intersect at their second stop, and one checkpoint defines the requirements for each line. Each line only covers one direction and provides 1 h of service. Hence, a total of 120 min of service time is covered.

Instances I2 and I3 refer to the same PTN with three bi-directional lines. Each line covers five stops. Lines two and three intersect the first line and enable passenger transfers. A checkpoint is positioned right after the start of each line. Hence, a total of six checkpoints set requirements toward each direction of each line. Instance I2 covers 2 h of service time for each line in each direction. A total of 720 min (60 min * 2 h * 3 lines * 2 directions) of service time is covered. Instance I3 comprises the same PTN as instance I2, but each line serves 8 h of service time, a total of 2880 min.

The PTN of I4 and I5 comprises five bi-directional lines and a total of 29 stops. Lines one and two cover the same amount of stops and run parallel for their remaining routes after intersecting. The third line additionally covers the stops where the first and second lines run parallel. Line four covers nine stops and intersects with the first three lines at the same stop. The passengers can connect to either of the three lines depending on their preferences. Line five covers three stops, and its last stopping point intersects with the first line. Importantly, some checkpoints set requirements toward multiple lines simultaneously. Hence, the set of trips from different lines passing the checkpoint must not violate the minimal headway, maximal headway, and passenger requirement constraints. A total of 11 checkpoints define the requirements for a total of 54 time periods. Instance I4 covers 195 h (11,700 min), with each line covering about 20 h of service time for every direction. To evaluate the impact of an increased service time on the runtime of the proposed solution approaches, instance I5 corresponds to I4 but covers a fivefold service time, and each time period within every checkpoint is five times longer. Lastly, to examine the influence of increasing the PTN size on the runtime, I6 comprises five distinct PTNs according to I4. In order not to significantly increase the PTNs' complexity of I6 compared to I4, deadheading between each duplicated PTN is not allowed.

The same cost structure is applied to each approach utilized in this paper. Each kilometer a bus travels in a trip, deadhead, pull-in, or pull-out costs 1.5 units. Each minute a bus is outside of the depot costs an additional 0.5 units. The costs and capacity of a bus depend on its vehicle type. The basic bus type has a capacity of 80 passengers and costs 10,000 units. Double-decker buses transport 120 passengers and cost 18,000 units. Mini-buses transport 20 passengers and cost 5000 units. A connection counts as reached if the connecting line departs at least 2 min later ($\underline{w}_x = 2, \forall x \in X$) and not more than 10 min later ($\overline{w}_x = 10, \forall x \in X$). Preliminary experiments suggest a reward of minus five cost units for reaching a connection and minus one cost unit for each good headway. The latter bonus is granted when solving the TTVSP-HC′ for each clock-faced headway and the headways are additionally required to be regular in the scope of solving the TTVSP-HC. The runtime for every exact computation is limited to 24 h. For the AMEES, the runtime is limited to 5 min for solving the TTVSP and 10 min for solving the TTVSP-HC.

Table 3 Results of exactly sequentially solving the TT and VSP

Instance	#Trips	#Buses	Obj. value	Runtime (s)	
				TT	VSP
I1	8	6	60,872	0.10	0.09
I2	78	30	307,446	0.18	0.17
I3	318	30	318,636	0.46	0.87
I4	419	19	209,790	4.41	2.69
I5	2081	21	306,455	22.13	24.17
I6	2095	97	1,092,118	47.81	113.07

5.2 Improvements Due to the Integrated Solution of the TTVSP

In this section, we evaluate the advantages of an integrated optimization compared to a sequential approach. Each instance described in Section 5.1 is first solved sequentially and then integrated. We compare the results in terms of solution quality and runtime.

In a sequential approach, solving the TT aims to minimize the number of scheduled trips while every checkpoint requirement is met. Connections and good headways can be included to improve the timetables' quality further. To exclusively assess the impact on the nominal and operational costs of an integrated compared to a sequential approach, these additional quality criteria are not considered yet. The computed timetable serves as input for solving the corresponding VSP. Table 3 displays results of exactly sequentially solving the TT and VSP for instances I1 to I6. For each *Instance*, the column *#Trips* indicates the minimal number of trips required to meet the timetabling requirements. The column *#Buses* provides the number of buses necessary to serve the timetable. The *Objective Value* comprises the nominal costs for every vehicle and operational costs for scheduled trips, deadhead trips, and idle times. The last columns display the *Runtime in seconds* to compute the timetable and vehicle schedule sequentially.

The number of scheduled trips increases corresponding to the extended service time for the instances. However, the demand for buses does not necessarily correlate with the number of planned trips. Even though instance I3 covers the same PTN as I2 with an increased service time, the same fleet size is required to serve the timetable. Compared to the small instances I2 and I3, the structure of instance I4 appears to allow better deadheading. Hence, even if more trips are planned in total, fewer vehicles are required to serve the timetable. Instance I5 covers five times the service time of I4. In total, only two more vehicles are required. Since the PTN of I6 does not facilitate deadheading between the distinct PTNs associated with I4, a fleet size that is exactly five times larger than that of I4 is expected. However, even more vehicles are required. This indicates that a sequential approach does not lead to the overall best solution. The objective value is mainly influenced by the fleet size and further increases in the quantity of scheduled trips. Intuitively, the runtime for exactly solving the TT and VSP sequentially increases with the increasing complexity of the solved instance.

Compared to the sequential approach, the TTVSP covers a significantly increased solution space but enables the computation of the best overall solution. Table 4 displays the results of solving the TTVSP for every instance and is structured in a similar

Table 4 Results of solving the TTVSP (Δ-comparison to sequential results)

Instance	#Trips	#Buses	Objective value	#Clock headways	#Connections	Runtime (s)	Gap after 24 h
I1	9 (+1)	5 (−1)	50,787 (−19.86%)	1	3	0.30	-
I2	78 (+0)	27 (−3)	276,840 (−11.01%)	4	0	5.68	-
I3	318 (+0)	27 (−3)	287,037 (−11.01%)	6	23	197.10	-
I4	[464] (+45)	[39] (+20)	[658,608] (+70.23%)	64	70	24 h	48.3%
I5	–	–	–	–	–	24 h	-
I6	–	–	–	–	–	24 h	-

manner to Table 3. Additionally, the difference in trips, fleet size, and objective value compared to the sequential approach is displayed in brackets after the corresponding value. Headways and connections are not in the scope of solving the TTVSP. However, to evaluate the benefit of solving the TTVSP-HC, the columns *#Clock-Headways* and *#Connections* serve as a benchmark and display the quantity of clock-faced headways and connections reached. If no optimal solution was computed after 24 h runtime, the remaining optimality gap is displayed in the column *Gap after 24 h*. If a feasible, non-optimal solution was found within 24 h, squared brackets around the number of trips, buses, and the objective value indicate that they might not be optimal yet.

An exact integrated approach is capable of calculating optimal results for I1, I2, and I3 in a short computation time. In less than 24 h, a feasible solution was computed for I4, but not for I5 nor I6. Surprisingly, the solution for I1 required one less bus but scheduled an additional trip. This can only be the case if either serving a trip is faster than deadheading or if scheduling an additional trip leads to a more flexible timetable, hence better vehicle rotations. Since we defined deadheads always to be faster than a trip, the added trip led to more freedom in planning and decreased the fleet size. The solutions for both I2 and I3 scheduled the same amount of trips compared to the sequential approach. However, the integrated approach required significantly fewer vehicles and improved the objective value considerably. Solving the TTVSP already resulted in some clock-faced headways and a few reached connections.

Overall, a sequential approach is capable of solving the TT and VSP for real-world-inspired instances in reasonable computation time. However, a sequential approach will most likely not result in a minimal-cost solution. In comparison, an integrated approach significantly improves the overall solution for small instances. For medium to large real-world-inspired instances, an integrated optimization is not capable of computing optimal solutions in reasonable computation time and, as a result, is not applicable for real-world use.

5.3 Exactly Solving the TTVSP-HC′

Compared to solving the TT and VSP sequentially, the TTVSP covers a significantly increased solution space. By considering clock-faced headways and maximizing con-

Table 5 Results of exactly solving the TTVSP-HC (Δ-comparison to TTVSP)

Instance	#Trips	#Buses	Objective value (without bonus)	Bonus
I1-HC′	9 (+0)	5 (+0)	50788.5 (+1.5)	23
I2-HC′	78 (+0)	27 (+0)	276,840 (+0)	256
I3-HC′	[321] (+3)	[27](+0)	287,046 (+9)	1055
I4-HC′	[462] (−2)	[45] (+6)	[1,149,527] (+43%)	[333]
I5-HC′	–	–	–	–
I6-HC′	–	–	–	–
Instance	#Clock headways	#Connections	Runtime (s)	Gap after 24 h
I1-HC′	3 (+2)	4 (+2)	2.39	–
I2-HC′	6 (+4)	50 (+50)	144.27	–
I3-HC′	0 (−6)	211 (+188)	24 h	1.41%
I4-HC′	(60) (−4)	[33] (−47)	24 h	87.1%
I5-HC′	–	–	24 h	–
I6-HC′	–	–	24 h	–

nections, the TTVSP-HC′ further increases the problem's complexity. As described in Section 3, the exact formulation excludes regularity and maximizes clock-faced headways only. Evaluating the solution quality, runtime, and benefit of considering additional quality criteria, Table 5 displays the results of exactly solving every problem instance considered a TTVSP-HC′. The *-HC′* suffix after an instance's name indicates the consideration of clock-faced headways and connections. For each *Instance*, the column *#Trips* shows the scheduled number of trips to serve a timetable, and the column *#Buses* the minimum fleet size. To allow a detailed assessment of the impact of solving the TTVSP-HC′, the displayed *Objective Value* excludes bonuses from clock-faced headways and connections. The column *Bonus* comprises the gained bonuses from scheduling *#Clock-Headways* and reaching *# Connections*. The difference compared to the TTVSP solution is displayed in brackets behind each value. The last two columns show the *Runtime in seconds* and a possible *Optimality Gap after 24 h*. If no optimal solution was found within 24-h runtime, squared brackets indicate the current feasible but non-optimal value of the solution.

Optimal results have been obtained for I1-HC′ and I2-HC′ in less than 24 h. For I3-HC′ and I4-HC′, a feasible solution with a remaining optimality gap was calculated. For I5-HC′ and I6-HC′, no feasible solution was found in less than 24 h. Focusing on the two small instances solved to optimality first, the amount of trips and buses remain equal. However, the objective values rise slightly in favor of achieving bonuses. Deadhead and idle times are increased to improve the amount of both clock-faced headways and connections. Still, only a fraction of trips is planned in clock-faced intervals. Supposedly, additional trips have to be scheduled to allow a more flexible timetable and improved headways. However, no extra trips are planned if the increased nominal and operational costs outweigh the benefit of scheduling more trips and achieving a

higher clock-faced headway bonus. On the other hand, with only slightly increased operational costs, the quality of connections improves substantially. By increasing the service times of I2-HC′, an exact approach is already incapable of computing an optimal solution in less than 24 h for I3-HC′. The feasible solution schedules three trips more than the TTVSP timetable. The fleet size remains equal, and the objective value increases slightly. Even though fewer clock-faced headways are planned, the connections significantly improve and lead to a bonus outweighing the increased objective value. The large optimality gap for I4-HC′ does not enable further conclusions.

Overall, an exact approach for solving the TTVSP-HC′ further increased the solution quality in reasonable computation time for tiny to small instances. Marginally increased costs led to an overall quality improvement of the timetable. However, the proposed model is not applicable when solving the TTVSP-HC′ for medium or large real-world-inspired instances.

5.4 Utilizing the AMEES to Solve the TTVSP and TTVSP-HC

As shown in Sections 5.2 and 5.3, an exact approach is unable to solve the TTVSP or TTVSP-HC′ for medium to large real-world instances in reasonable computation time. To assess the limits, benefits, and practical capability of the AMEES, the TTVSP and TTVSP-HC are solved for instances I1(-HC) to I6(-HC) and the results are compared to those of the exact sequential and integrated approaches. In contrast to the exact approach, when solving the TTVSP-HC with the AMEES, regularity for headways is considered in addition to them being clock-faced. Experiments in this chapter always refer to the TTVSP-HC′ for exact computations and the TTVSP-HC when the AMEES is utilized.

The AMEES is executed 20 times for each instance. The runtime limit is set to 5 min for solving the TTVSP and 10 min for the TTVSP-HC. Table 6 displays the results of the best solution from every execution. For each *Instance*, the column *#Trips* gives information about the average, minimum, and maximum number of scheduled trips required to comply with the timetabling requirements. The presence of the suffix *-HC* after an instance's name indicates that the solution is considered a TTVSP-HC, and as a TTVSP otherwise. The column *#Buses* displays the average, minimum, and maximum required fleet size to serve the corresponding timetable. The average, minimum, and maximum objective value without bonuses from each execution is shown in the column *Obj. Value*. The column *Runtime (s)* indicates the average, minimum, and maximum time in seconds the AMEES required to compute the best solution. The average, minimum, and maximum scheduled number of *Good Headways* and *Connections* are shown in the last two columns.

The AMEES successfully computed solutions for every instance considered a TTVSP and TTVSP-HC in reasonable computation time. For each instance, the average required number of trips only slightly deviates from, or equals, the best solution from every run. Except for one execution of I5, each run resulted in the same fleet size. As a result, the objective value varies only within a small margin in each execution. In favor of increasing good headways and connections, the objective value of the TTVSP-HC is slightly higher compared to the instances solved as a TTVSP.

Table 6 Results of heuristically solving the TTVSP and TTVSP-HC

Instance	#Trips			#Buses			Obj. value (without bonuses)			Runtime (s)			#Good H.			#Connections		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
I1	9.25	9	10	5	5	5	50,796	50,787	50,825	0.93	0.08	3.10	1	1	1	3.2	2	5
I1-HC	9.90	9	10	5	5	5	50,825	50,824	50,827	5.09	1.24	9.7	3.2	3	4	4.9	4	5
I2	78	78	78	27	27	27	276,924	276,840	277,563	7.39	4.51	9.80	3.50	0	10	27.60	18	36
I2-HC	78.1	78	79	27	27	27	277,008	276,585	279,066	22.92	10.63	29.97	5.9	0	7	46.13	38	51
I3	318.2	318	319	27	27	27	287,097	287,032	287,332	14.36	2.40	19.77	19.6	0	57	100.6	40	156
I3-HC	319.9	318	321	27	27	27	287,188	287,097	287,282	63.7	13.12	88.63	4.3	0	9	189.8	157	211
I4	419	419	419	19	19	19	191,738	191,629	191,870	17.5	8.8	31.2	189.4	104	318	62.4	36	86
I4-HC	419	419	419	19	19	19	191,895	191,629	192,134	139	41	222	342.1	315	366	103.4	98	110
I5	2082	2081	2086	19	19	20	209,833	207,699	224,003	42.5	26.3	54.5	922.3	774	1399	226.2	180	426
I5-HC	2082	2081	2085	19	19	19	207,923	207,752	208,094	364	150	529	1484.9	1243	1720	489.0	334	525
I6	2095	2095	2096	95	95	95	958,195	958,145	959,145	32.5	12.1	–	873.0	673	1206	318.4	230	495
I6-HC	2095	2095	2097	95	95	95	958,964	958,145	959,772	311	179	446	1205.6	1308	1630	503.1	462	550

Since scheduling more trips enables the possibility for more good headways and connections, the TTVSP-HC solution schedules more trips on average compared to the corresponding TTVSP solution. Overall, the runtime is reasonable, increases with the instances' complexity, and is further extended when considering good headways and connections. The TTVSP-HC improves the amount of good headways compared to the TTVSP solution. Interestingly, the solution for I3 schedules more good headways on average compared to I3-HC. This individual case can be explained by further examining I3-HC's solution. Compared to I3, significantly more connections are scheduled. Since connections yield a considerably higher bonus, they are preferred to be scheduled at the cost of increasing good headways. Likewise, the connections are notably increased for each TTVSP-HC solution. Since the evaluation of every MO requires some initial time in each execution, both the increased service time of I5 and the increased PTN of I6 exhibit the benefit of properly weighted and appropriately applied MOs. With increasing runtime, MOs are applied less randomly and frequently improve the solution. Hence, both instances only required nearly twice the runtime of I4. Only one execution of I6 did not lead to a feasible solution (indicated by a missing maximal runtime value). Even though the small instances I1(-HC), I2(-HC), and I3(-HC) appear to be solved more easily compared to larger instances, the small PTNs and the few scheduled trips facilitate only a small range of flexibility. Hence, only a few very specific combinations of trips lead to a cost minimal solution and are challenging to compute. The larger instances I4(-HC), I5(-HC), and I6(-HC) offer more degrees of freedom. Several different arrangements of scheduled trips in the timetable result in the same objective value. Therefore, the same or a similar solution quality can be achieved frequently.

The average solution value computed by utilizing the AMEES is compared with those of the sequential and exact approaches in Table 7. For each *Instance*, the columns *#Trips*, *#Buses*, and *Objective Value* (without bonuses) display the average result of the AMEES and the difference to the sequential approach (Table 3) and exact approach (Tables 4 and 5). For instances that do not consider bonuses for headways and connections, a bonus was not computed and the corresponding cell is labeled with "N/A" (not applicable). To evaluate the quality of the AMEES and the benefits of considering bonuses, the number of good headways and connections is also displayed for instances that did not explicitly consider bonuses in the solution process. Cells containing only a dash indicate that no corresponding value was computed within the time limit. For the TTVSP-HC, the column *Bonus* compares the received bonuses from scheduling good headways and connections utilizing the AMEES to the exact computation. Lastly, the computed *#Headways* and *#Connections* leading to the bonus are evaluated.

Compared to the sequential approach, on average, the AMEES scheduled slightly more trips. However, except for I4, significantly fewer vehicles are required. The average objective value is considerably lower for each instance. As a result, utilizing the AMEES to solve the TTVSP and TTVSP-HC is preferred to applying a sequential approach. Compared to the exact computation of the TTVSP and TTVSP-HC, the AMEES scheduled up to two additional trips on average for I1(-HC), I2(-HC), and I3(-HC). For these instances, the AMEES resulted in the same minimal fleet size. On average, the objective value is slightly higher. Since the AMEES scheduled on average one additional trip for I1-HC, more good headways and connections can be scheduled.

Table 7 Comparison of AMEES TTVSP, TTVSP-HC to sequential and exact

Instance	#Trips			#Buses			Obj. value (without bonuses)			Bonus		#Good H.		#Connections	
	Avg	Δ Seq	Δ Exact	Avg	Δ Seq	Δ Exact	Avg	Δ Seq	Δ Exact	Avg	Δ Exact	Avg	Δ Exact	Avg	Δ Exact
I1	9.25	+1.25	+0.25	5	−1	+0	50,796	−10,076	+10	N/A	N/A	1	+0	3.2	+0.2
I1-HC	9.90	+1.9	+0.9	5	−1	+0	50,825	−10,047	+37	27.7	+4.7	3.2	+0.2	4.9	+0.9
I2	78	+0	+0	27	−3	+0	276,924	−30,522	+85	N/A	N/A	3.5	−0.5	27.6	+27.6
I2-HC	78.1	+0.1	+0.1	27	−3	+0	277,008	−30,438	+168	236.55	−19.45	5.9	−0.1	46.13	−3.9
I3	318.2	+0.2	+0.2	27	−3	+0	287,097	−31,539	+61	N/A	N/A	19.6	+13.6	100.6	+76.4
I3-HC	319.9	+1.9	+1.9	27	−3	+0	287,188	−31,448	+142	953.8	−101.2	4.3	+4.3	189.8	−21.2
I4	419	+0	–	19	+0	–	191,738	−18,270	–	N/A	N/A	189.4	–	62.4	–
I4-HC	419	+0	–	19	+0	–	191,895	−17,894	–	859.1	–	342.1	–	103.4	–
I5	2082	+1	–	19	−2	–	209,833	−96,612	–	N/A	N/A	922.3	–	226.2	–
I5-HC	2082	+1	–	19	−2	–	207,923	−98,532	–	3929.9	–	1484.9	–	489.0	–
I6	2095	+0	–	95	−2	–	958,195	−133,923	–	N/A	N/A	873.0	–	318.4	–
I6-HC	2095	+0	–	95	−2	–	958,964	−133,154	–	3721.1	–	1205.6	–	503.1	–

Hence, the bonus is increased. For I2(-HC) and I3(-HC), a lower bonus is achieved on average. However, the AMEES is capable of solving I4(-HC), I5(-HC), and I6(-HC) in reasonable computation time. An exact approach did not compute an optimal solution within 24 h. As a result, the AMEES is preferred over an exact approach to solving the TTVSP and TTVSP-HC for instances of increased complexity.

The AMEES is capable of computing TTVSP and TTVSP-HC solutions in a short computational time. Compared to a sequential approach, the objective value is consistently decreased. Hence, the AMEES is strictly preferred to a sequential approach. For small instances, an integrated exact approach results in a lower objective value on average. However, the solutions obtained by the AMEES deviated only slightly within each execution and led at least once to the best overall solution in 20 runs. Additionally, the AMEES outperforms an exact TTVSP and TTVSP-HC approach for complex real-world-inspired instances. Within a reasonable time, the AMEES yields a high-quality timetable and vehicle schedule with an overall lower objective value than a sequentially computed solution. Overall, the AMEES exceeds the limits of an exact integrated approach and is applicable to differently sized real-world-inspired instances of varying complexity.

5.5 Convergence Behavior of the AMEES

As stated in Section 4.2, the AMEES attempts to compute a valid solution first, followed by the minimization of the total costs. To achieve either goal, each individual MO contributes to the solution process. Table 8 displays how often each proposed MO *Class* was *Applied* in two exemplary executions of solving the real-world-inspired instance I4. The column *Improved* exhibits how often the applied MO class led to an immediately improved solution. For the first run (A), in total, 4,859,415 MOs have been applied, leading to 122,116 overall immediate improvements. The second run

Table 8 MO statistic for solving I4-HC

Scope	Class	Run A Applied	Improved	Run B Applied	Improved
TT	T-Planner	264,776	51,672	274,876	58,308
TT	T-Unplanner	114,129	6440	120,413	5401
TT	T-Switcher	157,306	1191	157,592	862
TT	T-Allocator	437,060	47,355	516,185	56,949
VSP	V-Planner	3,289,292	0	3,372,593	0
VSP	V-Merger	263,361	10,628	229,953	4499
VSP	V-Splitter	49,018	0	76,986	0
VSP	V-Allocator	120,790	2976	105,429	400
TTVSP	Efficient-VehicleRotation	65,887	1807	53,847	157
TTVSP	VehicleReducer	24,995	6	26,723	23
TTVSP	VehicleTrip-Smoother	47,806	40	51,472	49
Total:		4,859,415	122,116	4,986,069	126,648

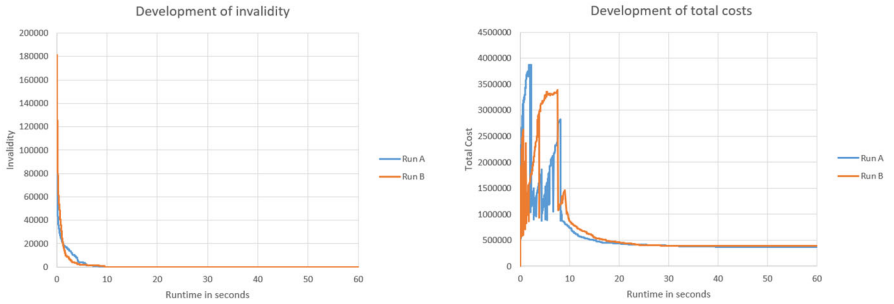


Fig. 5 Convergence behavior of two runs solving I4

(B) applied 4,986,069 MOs, which led to 126,648 immediate improvements. Note that these numbers reflect the total number of applications and improvements. The final best solution of Run A required 1,632 MOs, and Run B applied 1,437 MOs. To solve the TTVSP, each class was applied frequently and jointly led to computing a high-quality solution in either run. Since trips must be assigned to buses in the AMEES at all times, every scheduling and a notable amount of switching and reallocation of trips requires planning a new bus. As a result, the V-Planner MO was applied most. However, since scheduling a bus always increases the objective value, the class never led to an immediate improvement. Nevertheless, the V-Planner MO is mandatory to compute feasible solutions and, hence, is applied frequently. Similarly, the V-Splitter MOs always lead to an increased fleet size, but they enable further improvements of the solution by subsequently applying another MO.

For Runs A and B, the left side of Fig. 5 displays the development of the invalidity (y-axis) of the best solution at the current runtime (x-axis). In 10 s, a valid solution was computed in both runs. At this point, a sufficient number of passengers can be transported at each checkpoint, and both minimal and maximal headway conditions are met. Since achieving validity is prioritized first, the development of total costs shown on the right side of Fig. 5 behave differently. The y-axis displays the total operational and fixed costs of the best solution at the current runtime. In the beginning, the total costs equal zero since no bus is required if no service trips are scheduled. However, service trips are required to obtain a feasible solution. Hence, operational and fixed costs increase within the runtime. At certain points in time, the costs exhibit sudden decreases. This is the case because a different individual with a different cost structure yields the lowest invalidity. This high variance of total costs among different individuals arises from the continuous evaluation of MOs. While the primary objective is to attain validity, MOs aimed at enhancing operational efficiency and reducing costs are also applied. Occasionally, there are instances where an individual with lower total costs simultaneously achieves the lowest level of invalidity. Nevertheless, the validity of this particular individual may not be further improved, and a different individual with a higher cost structure is evaluated as the overall best. Consequently, the total costs increase, and the graph jumps back to its previous level. Finally, after validity is achieved, the total costs are minimized and improve steadily in the remaining runtime until both runs converge to the same value.

To facilitate the significance of interdependence among various MO classes, Fig. 6 in Appendix 12 displays the development of the invalidity (left side) and total costs (right side) for different compositions of MO classes. Every diagram in Fig. 6 displays five independent exemplary runs for 60 s of runtime. The first diagrams (Fig. 6a) serve as the benchmark for each other composition and display the development of invalidity and total costs when incorporating every MO class. Each run computes a valid solution quickly and eliminates every constraint violation by 11 s at the latest. The operational and fixed costs of each of the runs demonstrate a progression comparable to that of Runs A and B and converge to the same minimal value after achieving validity. Diagrams (Fig. 6b–i) display a set of independent runs, each excluding one specific class of MOs for solving the TTVSP (I4). In what follows, we will briefly describe the most crucial difference between each composition and the benchmark (Fig. 6a). Diagram (Fig. 6b) displays the development of invalidity and costs when excluding every T-Planner MO. Interestingly, no valid solution can be computed in either run. Since the primary goal of computing valid solutions is not achieved, the operation costs do not converge either. In contrast, when excluding T-Unplanner (Fig. 6c), validity is achieved as quickly as in the benchmark scenario. However, since unnecessary trips can not be efficiently removed from vehicle rotations, the operational costs converge significantly slower and less reliably. A similar behavior can be observed when excluding T-Switcher (Fig. 6d). Validity is achieved quickly, but the operational costs are reduced even more slowly. Excluding the class of T-Allocator (Fig. 6e) does not influence the convergence of operational costs, as is the case for the two prior compositions. However, the validity is achieved less reliably but still as fast as in the benchmark scenario.

Since MOs in the scope of the VSP do not have a significant impact on achieving validity, excluding either of the four classes does not impact the development of invalidity significantly (Fig. 6f–h). For I4, a minimal fleet size of 27 buses is required in the minimal-cost solution. Even though every run, when excluding V-Merger (Fig. 6f), does converge to a similar value, neither of the five runs resulted in less than 33 vehicles. The exclusion of V-Splitter (Fig. 6g) resulted in the smallest fleet sizes compared to excluding V-Merger and V-Allocator, but still required a significantly larger fleet size of 30 vehicles on average. For computing the minimal fleet size, the V-Allocator (Fig. 6h) has the highest impact. The total costs converge to a significantly higher value than in any other composition that achieved validity and required more than 53 vehicles in every run.

Excluding the class of TTVSP MOs (Fig. 6i) exhibits a different behavior than any other composition. In comparison, the cost development of diagram (Fig. 6i) is the only one that does not exhibit jumps before achieving validity. Presumably, these jumps can be traced back to the TTVSP-class MOs. Thus, integrated MOs can lead to the computation of individuals, improving both validity and costs significantly, even before achieving validity.

In summary, the AMEES utilizes a variety of MOs to compute and improve a TTVSP(-HC) solution. Even though some MOs do not lead to an improvement immediately, each MO class facilitates finding a solution in the long term. For different runs, the quantity and sequence of applied MOs vary. As illustrated in Fig. 6 in Appendix 12, it is crucial to utilize the entirety of MOs and apply them adaptively to quickly achieve high-quality solutions and ensure reliable convergence toward similar objective values.

6 Conclusion and Outlook

In this work, we have investigated the TTVSP-HC and evaluated the applicability of an exact sequential, exact integrated, and heuristic solution approach. The benefits of considering good headways and connections are evaluated, and the limitations of exactly solving the TTVSP and TTVSP-HC′ are assessed. We propose an AMEES (adaptive modular evolutionary extendable scheme) capable of solving large real-world-inspired TTVSP-HC instances considering multiple connections, vehicle types, depots, and interlining of vehicles.

Comparing the results of an exact sequential and exact integrated approach demonstrates the substantial advantages of integration. Compared to solving the TT first and using the computed timetable as an input for the VSP, solving the TTVSP reduces the overall costs significantly. When considering clock-faced headways and connections, the TTVSP-HC′ further increases the solution quality. However, neither for a TTVSP nor TTVSP-HC′ is the proposed exact approach capable of computing optimal solutions for medium or large real-world-inspired instances in a reasonable time. Future research could investigate whether different MIP formulations accomplish superior results and reduce computational time. Further, instead of considering flexible headways, assuming fixed headways for different time periods could allow for the utilization of different, more efficient approaches applicable to cyclic timetable computations only. However, since already solving the TT considering synchronization is classified as NP-hard, given the current technological capabilities, it is not expected that any exact approach is suitable for real-world TTVSP-HC′ applications yet.

The proposed AMEES enables the possibility of solving the TTVSP and TTVSP-HC for instances of increased complexity and size. The computed solutions outperform the sequential approach for every considered instance. Even though an exact integration yields slightly better results for small instances on average, the inability to consider real-world PTNs prevents a practical application. Further, the AMEES is capable of considering regularity in addition to clock-faced headways and computes high-quality solutions for the TTVSP-HC in reasonable computational time. As a result, utilizing the AMEES or a similar heuristic for solving the TTVSP and TTVSP-HC is strongly encouraged in a real-world environment.

The use of AMEES to support planning offers various advantages for public transport companies. Due to the short computation time, the obtained solutions can be used diversely. They can, for example, be used as a starting point or benchmark for experienced planners. Additionally, possible improvements for manually or sequentially planned solutions can be evaluated. The AMEES converges to high-quality solutions with a similar objective value frequently. The promising results of solving the TTVSP and TTVSP-HC suggest a reliable heuristic scheme. However, the computed solution might still not be optimal yet and offers further room for improvement.

Each MO pursues its own task and applies small, predictable changes in a comprehensive way. The order of applying each MO can be traced for each solution. Therefore, the AMEES yields explainable results and can be further improved. The performance of the AMEES strongly depends on the composition of purposeful, efficient MOs. To consistently achieve high-quality results, problem-specific knowledge is required to design an efficient pool of various MOs. If an experienced planner identifies beneficial

changes to the timetable or vehicle rotations, the AMEES can be extended with a new MO instructed with this task only. This modular structure enables further improvements in runtimes and the solution quality without interfering with other aspects of the scheme.

Since the solution manager (AOC) is not problem-specific, a similar structure as the AMEES can potentially be utilized to solve problems from other domains too. For any problem that is restricted by time and is such that small gradual improvements can lead to a desired solution quality, utilizing the AOC or a similar solution manager in a scheme oriented to the AMEES might achieve promising results and can be examined in future research. An evaluation of runtimes of exact approaches can be decreased when including a feasible solution obtained by the AMEES might offer additional new insights. Further, the impact of applying the AMEES to larger PTNs or a sequence of days can be assessed. We aim to extend the AMEES by integrating crew scheduling as a third planning step and evaluate the limits and advantages of a threefold integration.

Appendix 1. Sets, Parameters, and Variables

Table 9 Sets, parameters, and variables of model TTVSP-HC

Sets	
A	All arcs
A^v	All arcs associated with a vehicle of type $v \in V$
A^v_c	All circulation flow arcs
A^v_d	All deadhead trip arcs $d^v \in D^v$
A^v_i	All idle time arcs $i_v \in I^v$
$A^v_{p^{out}}$	All pull-out trip arcs $p^v_{out} \in P^v_{out}$
$A^v_{p^{in}}$	All pull-in trip arcs $p^v_{in} \in P^v_{in}$
A^v_s	All service trip arcs $s^v \in S^v$
AC_x	Tuple of trip arcs $(a_1, a_2) \in A^v_s$ that intersect at $x \in X$ and are such that a timely connection is achieved
AH_o	Tuple of trip arcs $(a_1, a_2) \in A^v_s$ that depart in a clock-faced interval at stop $o \in HO$
$\underline{AH}^v_c$	For each $v \in V$ and $c \in C$, contains subsets $\underline{AH}^v_{c,ct}$ that only allow one service trip to be scheduled
$\overline{AH}^v_c$	For each $v \in V$ and $c \in C$, contains subsets $\overline{AH}^v_{c,ct}$ that require at least one service trip to be scheduled
$\underline{AH}^v_{c,ct}$	If multiple service trip arcs $a^v_s \in A^v_s$ in this set are scheduled, the minimal headway is violated
$\overline{AH}^v_{c,ct}$	If no service trip arc $a^v_s \in A^v_s$ in this set is scheduled, the maximal headway is violated
AP^v_c	Set of service trip arcs $a^v_s \in A^v_s$ that pass each $c \in C$
C	All virtual demand checkpoints
CT_c	All covered time horizons as sets of timestamps in minutes for each checkpoint $c \in C$
D^v	All possible deadhead trips carried out by a vehicle of type $v \in V$

Table 9 continued

Sets	
H^{clock}	Every time interval that is a factor of 60 and should be rewarded
HO	Every stop $o \in O$ where clock-faced headways should be rewarded
I^v	All possible idle times by a vehicle of type $v \in V$
L	All lines
N	All nodes
N^v	All nodes associated with a vehicle of type $v \in V$
O	All stops in the PTN
P^v_{out}	All possible pull-out trips carried out by a vehicle of type $v \in V$
P^v_{in}	All possible pull-in trips carried out by a vehicle of type $v \in V$
R	All routes
S	All possible service trips
S^v	All possible service trips carried out by a vehicle of type $v \in V$
T_l	All service time horizons as sets of timestamps in minutes for each line $l \in L$
U	All directions
V	All vehicle type and depot combinations
X	Tuple of every two routes r^u_l that intersect and for which well-timed passenger transfers are desired
Parameters	
α	Weighs the impact of rewarding connections
β	Weighs the impact of rewarding clock-faced headways
cap_v	Passenger capacity of a vehicle of type $v \in V$
co_c	Tuple of stops $o \in O$ where each checkpoint $c \in C$ is positioned
$\underline{ct}_c$	First timestamp in CT_c by each $c \in C$
$\overline{ct}_c$	Last timestamp in CT_c by each $c \in C$
$\underline{h}_c$	Minimal headway in minutes for each $c \in C$
$\overline{h}_c$	Maximal headway in minutes for each $c \in C$
M	Equals the highest value that the function $HC(o, a_1, a_2)$ can possibly return
$\underline{p}_c$	Minimal amount of passengers to be transported within CT_c for each $c \in C$
r^u_l	A route that covers a unique ordered subset of stops $o \in O$
$\underline{r}^u_l$	First stop of route r^u_l
$\overline{r}^u_l$	Last stop of route r^u_l
$\underline{w}_x$	Time it takes for a passenger to arrive at the connection route in connection $x \in X$
$\overline{w}_x$	Upper waiting time limit to reward a connection $x \in X$

Table 9 continued

Functions	
$cost(arc)$	Returns type-, distance-, and duration-dependent costs for each arc $a \in A$
$HC(o, a_1, a_2)$	Returns every trip arc $a_s^v \in A_s^v$ that can possibly be scheduled between two service trips $a_1, a_2 \in A_s^v$ at stop $o \in HO$
$R(l, u)$	Returns a route r_l^u given a line and direction
$TT(r_l^u, o_1, o_2)$	Returns total travel time between two stops o_1, o_2 for a route r_l^u
Variables	
π_{a^v}	Equals one if service trip arc $a^v \in A_s^v$, pull-out trip $a^v \in A_{p^{out}}^v$, or pull-in trip $a^v \in A_{p^{in}}^v$ is scheduled, and zero otherwise
σ_{a^v}	Equals the flow value on deadhead arcs $a^v \in A_d^v$ and idle arcs $a^v \in A_i^v$
θ_{a^v}	Equals the flow of the outgoing circulation flow arc $a \in A_c^v$ and the demand for vehicles of type $v \in V$
b_{ac}^x	Equals one if a both trip arcs $(a_1, a_2) \in AC_x$ are scheduled, and zero otherwise
b_{ac}^h	Equals one if trip arcs $(a_1, a_2) \in AH_o$ are scheduled, and no trip arc $a \in HC(o, a_1, a_2)$ is planned, and zero otherwise

Appendix 2. Additional Sets, Functions, and Parameters for MOs

Table 10 Additional sets and functions for MOs

Sets	
E	Set of all vehicle rotations
AP_c	Set of service trip arcs $a_s^v \in A_s^v$ that pass each $c \in C$ for any vehicle type
Functions	
add(e)	Adds a new vehicle rotation e to E
$AH(a_s^v, i, c, b)$	Given an arc $a_s^v \in A_s^v$, a time interval i, a checkpoint $c \in C$, and a Boolean b, returns the next service trip with a headway i of the same line. If b equals FALSE and multiple lines pass the same checkpoint $c \in C$, instead returns a service trip with headway i from a randomly selected different line that passes the checkpoint
bestSplit(e)	Returns the lowest cost combination of two vehicles covering every service trip associated with vehicle $e \in E$
canMerge(e^1, e^2)	Returns TRUE if the service trips of two vehicles e^1, e^2 are not overlapping and they can be connected by deadhead trips, and FALSE otherwise
createVehicle(a_s^v)	Creates a new vehicle rotation e and adds it to E. If a service trip arc $a_s^v \in A_s^v$ is passed as an argument, the new vehicle covers service trip a, and necessary pull-out and pull-in arcs are added to e
first(S)	Given any set S, returns the first element from S
fitsInto(a_s^v, e)	Returns TRUE if adding service trip $a_s^v \in A_s^v$ to vehicle rotation e leads to a valid vehicle rotation

Table 10 continued

Functions	
following(a_s^v,c,b)	Given an arc $a_s^v \in A_s^v$, a checkpoint $c \in C$, and a Boolean b, returns the next scheduled service trip departing after a_s^v from the same line if b equals TRUE. If b equals FALSE and multiple lines pass the same checkpoint c, returns the first following scheduled service trip of a line different than the line associated with a_s^v
$\underline{HW}(a_s^v, c)$	Returns the headway of an arc $a_s^v \in A_s^v$ from the prior departing trip at $c \in C$. If there is no prior departure, returns NULL
$\overline{HW}(a_s^v, c)$	Returns the headway of an arc $a_s^v \in A_s^v$ to the next departing trip at $c \in C$. If there is no next departure, returns NULL
last(s)	Given any set s, returns the last element from s
linesPerC(c)	Returns the number of lines that pass a $c \in C$
nearest(i,S)	Returns the nearest integer to a given integer i from a set S of integers
merge(e^1, e^2)	Merges two vehicle rotations and returns the newly merged vehicle rotation
next(a_s^v,h)	Given a service trip arc $a_s^v \in A_s^v$, returns a service trip arc $a_s^v \in A_s^v$ with the departure time t_l being h minutes later
numElem(S)	Given any set S, returns the number of elements in S
prior(a_s^v,c,b)	Given an arc $a_s^v \in A_s^v$, a checkpoint $c \in C$, and a Boolean b, returns the prior scheduled service trip departing before a_s^v from the same line. If b equals FALSE and multiple lines pass the same checkpoint c, returns the prior scheduled service trip from a randomly selected different line that passes the checkpoint
$R'(a)$	Given a service trip arc a, returns the associated route r_l^u
randomFrom(S)	Given any set S, returns a random element from S
remove(e)	Removes a vehicle rotation e from E
scheduled(A)	Given any service trip arc set A, returns every scheduled trip with $\pi_a = 1$ for each $a \in A$
Vehicle-dependent functions, e $\in E$	
e.addTrip(a_s^v)	Adds a service trip $a_s^v \in A_s^v$ and the required deadhead and idle time arcs to vehicle e
e.covers(a_s^v)	For e in E, returns TRUE if arc $a_s^v \in A_s^v$ is covered by e
e.getC()	Returns the total operational and fixed costs for vehicle e
e.getCD()	Returns the total cost for deadheads of vehicle e
e.getCI()	Returns the total cost for idle times of vehicle e
e.getT()	Returns the total time for the vehicle rotation of vehicle e
e.getTrips()	Returns every service trip $a_s^v \in A_s^v$ covered by vehicle e
e.removeTrip(a_s^v)	Removes a service trip $a_s^v \in A_s^v$ and the associated deadhead and idle time arcs from vehicle e

Appendix 3. MO Class T-Planner

Table 11 Overview and concept of proposed T-Planner class

Name	Concept
MaxHeadway-Violated	For each $c \in C$, if the headway between two trips exceeds the maximal headway, a new trip is planned with a maximal headway to the first trip
MaxHeadway-ViolatedDiffLine	For each $c \in C$ that gets passed by multiple lines, if the headway between two trips exceeds the maximal headway, a new trip is planned from a different line with a maximal headway to the first trip
MinHeadway	For a random demand $c \in C$, plans a trip with a headway equal to the minimal headway if the trip has no successor
PassengerMissing	For each $c \in C$, if not enough trips are scheduled to transport all passengers, a new random trip associated with c is scheduled
Regular	For a random checkpoint $c \in C$, selects a random planned trip, that has no successor, and plans a new trip with the same headway as the headway to the previous departure at c
Connection	For a random tuple of trip arcs in AC, if one service trip is planned, plans the other one too
Clockheadway	For a random scheduled trip, plans a new trip with a random clock-faced interval
ServiceTime	For each $c \in C$ that is designed to ensure the service time[a], plans a random trip to cover the service time start or end if no trip is planned yet

[a]For checkpoints designed to ensure the service time of a line, $\overline{ct}_c$ equals the service time start, and $\overline{h}_c$ the margin of the first bus departure. If there is no minimal headway, we write $\underline{h}_c = \emptyset$

Listing 1 MaxHeadwayViolated

```
for c in C
 for a in scheduled(APc)
  if HW(a,c)==NULL
  or HW(a,c)>hc
   a*=AH(a,hc,c,TRUE)
   πa*=1
    exit MO
```

Listing 2 MaxHeadwayViolatedDiffLine

```
for c in C
 if linesPerC(c) > 1
  for a in scheduled(AP_c)
   if overline(HW)(a,c)==NULL
   or overline(HW)(a,c)>overline(h)_c
     a*=AH(a,overline(h)_c,c,FALSE)
     π_a*=1
       exit MO
```

Listing 3 MinHeadway

```
c=randomFrom(C)
for a in scheduled(AP_c)
 if overline(HW)(a,c)==NULL
  a*=AH(a,underline(h)_c,c,TRUE)
  π_a*=1
   exit MO
```

Listing 4 PassengerMissing

```
for c in C
 transported=0
 for v in V
  for a_c^v in AP_c^v
   if π_(a_c^v)==1
     transported+=cap_v
 if transported < underline(p)_c
  a*=randomFrom(AP_c^v)
  π_a*=1
   exit MO
```

Listing 5 Regular

```
c=randomFrom(C)
A=scheduled(AP_c)
for a in A
 if overline(HW)(a,c)==NULL
 and underline(HW)(a,c)!=NULL
  h*=underline(HW)(a,c)
  a*=AH(a,h*,c,TRUE)
  π_a*=1
   exit MO
```

Listing 6 Connection

```
(a^1, a^2) = randomFrom(AC)
if π_{a^1} == 1 xor π_{a^2} == 1
   π_{a^1} = 1
   π_{a^2} = 1
```

Listing 7 ServiceTime

```
for c in C
 if h_c == ∅
  A = scheduled(AP_c)
  if numElem(A) == 0
   a* = randomFrom(AP_c)
   π_{a*} = 1
   exit MO
```

Listing 8 Clockheadway

```
c = randomFrom(C)
A = scheduled(AP_c)
a = randomFrom(A)
h = randomFrom(H^{clock})
a* = AH(a, h, c, TRUE)
π_{a*} = 1
```

Appendix 4. MO Class T-Unplanner

Table 12 Overview and concept of proposed T-Unplanner class

Name	Concept
MinHeadway-Violated	For each $c \in C$, unschedules a service trip if two trips have a headway lower than the allowed minimal headway
Passenger-Overplanned	For a random $c \in C$, if more trips are planned than required to transport all passengers, unschedules the first service trip that does not violate the maximal headway constraint when being unscheduled

Listing 9 MinHeadwayViolated

```
for c in C
 for a in scheduled(AP_c^v)
  if HW(a, c) != NULL
   if HW(a, c) < h_c
    π_a = 0
    exit MO
```

Listing 10 PassengerOverplanned

```
c=randomFrom(C)
transported=0
for v in V
 for a_c^v in scheduled(AP_c^v)
  transported+=cap_v
if transported > p_c
 A=scheduled(AP_c^v)
 for a in A
  if HW(a)==NULL
  or HW(a)==NULL
   π_a=0
   exit MO
  if HW(a)
    + HW(a) ≤ h_c
    π_a=0
    exit MO
```

Appendix 5. MO Class T-Switcher

Table 13 Overview and concept of proposed T-Switcher MOs

Name	Concept
ServiceTime-StartLatest	Plans a trip as late as possible by still ensuring the service time of a line. Unschedules any other trip that passes a checkpoint designed for guaranteeing the service time[a]
ServiceTime-EndEarliest	Plans a trip as early as possible by still ensuring the service time of a line. Unschedules any other trip that passes a checkpoint designed for guaranteeing the service time[a]
MinHeadway-Violated	For each $c \in C$, if the minimal headway is violated, unplans a trip and schedules a later trip to increase the headway to the minimal headway
MaxHeadway-Violated	For each $c \in C$, if the maximal headway is violated, unplans a trip and plans an earlier trip to decrease the headway to the maximal headway
ClockHeadway	For a random planned trip, if the headway to the previous trip is not a clock-faced headway, unplans the trip and plans a new one with a clock-faced headway
Connection	For a random connection $x \in X$, selects a random tuple from $(a^1, a^2) \in AC_x$ that has only one trip scheduled, unplans the next/prior service trip and schedules the other trip in the connection

[a]For checkpoints designed to ensure the service time of a line, $\overline{ct}_c$ equals the service time start, and $\overline{h}_c$ the margin of the first bus departure. If there is no minimal headway, we write $\underline{h}_c = \emptyset$

Listing 11 ServiceTimeStartLatest

```
C* = ∅
for c in C:
 if h_c == ∅
  C*.addElem(c)
c = randomFrom(C*)
A = scheduled(AP_c)
a' = first(A)
if numElem(A) > 1
 π_a' == 0
 exit MO
if a' == last(AP_c)
 exit MO
if HW(a', c) == NULL
  a* = last(AP_c)
  π_a' = 0
  π_a* = 1
  exit MO
if HW(a', c) > h_c
 Δ = HW(a', c) - h_c
 a* = AH(a', Δ, c, TRUE)
 if a* not in AP_c
  a* = last(AP_c)
 π_a' = 0
 π_a* = 1
```

Listing 12 ServiceTimeEndEarliest

```
C* = ∅
for c in C:
 if h_c == ∅
  C*.addElem(c)
c = randomFrom(C*)
A = scheduled(AP_c^v)
a' = last(A)
if numElem(A) > 1
 π_a' == 0
 exit MO
if a' == first(AP)
 exit MO
if HW(a', c) == NULL
  a* = first(AP_c)
  π_a' = 0
  π_a* = 1
  exit MO
```

```
if HW(a', c) > h_c
 Δ = HW(a', c) - h_c
 a* = AH(a', −Δ, c, TRUE)
 if a* not in AP_c
  a* = first(AP_c)
 π_{a'} = 0
 π_{a*} = 1
```

Listing 13 MinheadwayViolated

```
for c in C
 A = scheduled(AP_c)
  for a in A
   if HW(a, c) < h_c
    Δ = h_c − HW(a, c)
    π_a = 0
    a* = HW(a, Δ, c, TRUE)
    π_{a*} = 1
    exit MO
```

Listing 14 MaxheadwayViolated

```
for c in C
 A = scheduled(AP_c)
  for a in A
   if HW(a, c) > h̄_c
    Δ = HW(a, c) − h̄_c
    π_a = 1
    a* = HW(a, -Δ, c, TRUE)
    π_{a*} = 0
    exit MO
```

Listing 15 ClockHeadway

```
c = randomFrom(C)
A = scheduled(AP_c)
a = randomFrom(A)
if HW(a, c) not in H^clock
 h = nearest(HW, H^clock)
 Δ = h - HW(a, c)
 a* = HW(a, Δ, c, 1)
 π_a = 0
 π_{a*} = 1
 exit MO
```

Listing 16 Connection

```
x=randomFrom(X)
AC_x^*=∅
for (a^1, a^2) in AC_x
 if π_{a^1}==1 xor π_{a^2}==1
  AC_x^*.addElement((a^1, a^2))
(a^1,a^2)=randomFrom(AC^*)
if π_{a^1}==1
  a'=following(a^1,c,FALSE)
  π_{a'}=0
  π_{a^2}=1
if π_{a^2}==1
  a'=prior(a^2,c,FALSE)
  π_{a'}=0
  π_{a^1}=1
```

Appendix 6. MO Class T-Allocator

Table 14 Overview and concept of proposed T-Allocator utilized in the AMEES for the TTVSP-HC

Name	Concept
MaxHeadway	Aiming to reduce maximal headway violations, multiple planned trips are unplanned and the same amount of trips are planned with decreased headways
MinHeadway	Aiming to reduce minimal headway violations, multiple planned trips are unplanned and the same amount of trips with increased headways are planned
Clockheadway	Plans and unplans multiple trips to have more headways be clock-faced headways
HeadwayIncreasing	Plans and unplans multiple trips to increase the headway to be the maximal headway
Regularity	Plans and unplans multiple trips to have more regular headways

Listing 17 MaxHeadway

```
c=randomFrom(C)
A=scheduled(AP_c)
Δ=0
for i=0 to numElem(A)-1
 if HW(A[i],c)+Δ > h̄_c
  Δ = HW(A[i],c)+Δ−h̄_c
  π_{A[i+1]}=0
  a^*=AH(A[i+1],-Δ,c,TRUE)
  π_{a^*}=1
 else
  Δ=0
```

Listing 18 MinHeadway

```
c=randomFrom(C)
A=scheduled(AP_c)
Δ=0
for i=0 to numElem(A)-1
 if HW̄(A[i],c) − Δ < h_c
    Δ = h_c − HW̄(A[i],c) + Δ
    π_{A[i+1]}=0
    a*=AH(A[i+1],Δ,c,TRUE)
    π_{a*}=1
 else
  Δ=0
```

Listing 19 Clockheadway

```
c=randomFrom(C)
A=scheduled(AP_c)
Δ=0
for i=0 to numElem(A)-1
 h=HW̄(A[i],c) + Δ
 if h not in H^clock
  h*=nearest(h, H^clock)
  Δ+= h* − HW̄(A[i],c)
 if Δ!=0
  π_{A[i+1]}=0
  a*=AH(A[i+1],Δ,c,TRUE)
  π_{a*}=1
```

Listing 20 Headwayincreasing

```
c=randomFrom(C)
A=scheduled(AP_c)
Δ = 0
for i=0 to numElem(A)-1
 if HW̄(A[i],c) − Δ < h̄_c
  Δ+= h̄_c − HW̄(A[i],c) + Δ
  π_{A[+1]}=0
  a*=AH(A[+1],Δ,c,TRUE)
  π_{a*}=1
 else
  Δ=0
```

Listing 21 Regularity

```
c=randomFrom(C)
A=scheduled(AP_c)
h=HW(A[0], c)
Δ=0
for i=1 to numElem(A)-1
 if HW(A[i], c)+Δ!=h
  Δ+=HW(A[i], c)+Δ-h
  π_A[i+1]=0
  a*=AH(A[i+1],Δ,c,TRUE)
  π_a*=1
 else
  Δ=0
```

Appendix 7. MO Class V-Planner

Table 15 Overview and concept of proposed V-Planner MOs utilized in the AMEES for the TTVSP-HC

Name	Concept
Vehicle	If a service trip $a_s^v \in A_s^v$ is planned ($\pi_{a_s^v} = 1$) and not yet assigned to a vehicle, creates a new vehicle and assigns the trip to it

Listing 22 Vehicle

```
A=scheduled(A_s^v)
for a in A
 isCovered=FALSE
 for e in E
  if e.covers(a)
   isCovered=TRUE
   exit for
 if not isCovered
  createVehicle(a)
```

Appendix 8. MO Class V-Merger

Table 16 Overview and concept of proposed V-Merger MOs utilized in the AMEES for the TTVSP-HC

Name	Concept
Extensive	Evaluates whether any two vehicle rotations can be merged and does so if possible
Best	Evaluates every possible merging of any two vehicle rotations and merges the rotations improving the solution quality most

Listing 23 Extensive

```
for e^1 in E
 for e^2 in E
  if canMerge(e^1, e^2)
   add(merge(e^1, e^2))
   remove(e^1)
   remove(e^2)
   exit MO
```

Listing 24 Best

```
Δ* = ∞
e* = ∅
ê^1 = ∅
ê^2 = ∅
for e^1 in E
 for e^2 in E
  if canMerge(e^1, e^2)
   cost = e^1.getC() + e^2.getC()
   e' = merge(e^1, e^2)
   Δ = cost - e'.getC()
   if Δ < Δ*
    Δ* = Δ
    e* = e'
    ê^1 = e^1
    ê^2 = e^2
if e* != ∅
   add(e*)
   remove(ê^1)
   remove(ê^2)
```

Appendix 9. MO Class V-Splitter

Table 17 Overview and concept of proposed V-Splitter MOs utilized in the AMEES for the TTVSP-HC

Name	Concept
BestFit	If a vehicle rotation can fit partially well into a different rotation, it is split and merged
HighCost	If a vehicle rotation has too many deadheads and too high idle costs, it is split into two rotations

Listing 25 BestFit

```
for e^1 in E
 for e^2 in E
  e^*1, e^*2=bestSplit(e^2)
  if canMerge(e^*1, e^1)
   add(merge (e^*1, e^1))
   remove(e^2)
   add(e^*2)
   exit MO
  if canMerge(e^*2, e^1)
   add(merge (e^*2, e^1))
   remove(e^2)
   add(e^*1)
   exit MO
```

Listing 26 HighCost

```
e=randomFrom(e)
c=e.getC()
c^i=e.getCI()
c^d=e.getCD()
if (c^i + c^d) > 0.2c
 e^*1, e^*2=bestSplit(e)
 add(e^*1)
 add(e^*2)
 remove(e)
```

Appendix 10. MO Class V-Allocator

Table 18 Overview and concept of proposed V-Allocator MOs utilized in the AMEES for the TTVSP-HC

Name	Concept
CostReduction	Allocates trips between vehicles to reduce the overall costs
TotatTime-Reduction	Allocates trips between vehicles to reduce the overall rotation times
Deadhead-Reduction	Allocates trips between vehicles to reduce the overall deadhead costs
IdleTime-Reduction	Allocates trips between vehicles to reduce the overall idle time costs

Listing 27 CostReduction

```
for e1 in E
 for e2 in E
  for a in e2.getTrips()
   if fitsInto(a, e1)
    cost=e1.getC()
         +e2.getC()
    e1.addTrip(a)
    e2.removeTrip(a)
    cost′=e1.getC()
         +e2.getC()
    if cost < cost′
     e1.removeTrip(a)
     e2.addTrip(a)
```

Listing 28 TotalTimeReduction

```
for e1 in E
 for e2 in E
  for a in e2.getTrips()
   if fitsInto(a, e1)
    t=e1.getT()
     +e2.getT()
    e1.addTrip(a)
    e2.removeTrip(a)
    t′=e1.getT()
     +e2.getT()
    if t < t′
     e1.removeTrip(a)
     e2.addTrip(a)
```

Listing 29 DeadheadReduction

```
for e1 in E
 for e2 in E
  for a in e2.getTrips()
   if fitsInto(a, e1)
    cd=e1.getCD()
       +e2.getCD()
    e1.addTrip(a)
    e2.removeTrip(a)
    cd′=e1.getCD()
       +e2.getCD()
    if cd < cd′
     e1.removeTrip(a)
     e2.addTrip(a)
```

Listing 30 IdleTimeReduction

```
for e1 in E
 for e2 in E
  for a in e2.getTrips()
   if fitsInto(a, e1)
    ci=e1.getCI()
       +e2.getCI()
    e1.addTrip(a_s^v)
    e2.removeTrip(a_s^v)
    it′=e1.getCI()
       +e2.getCI()
    if it < it′
     e1.removeTrip(a)
     e2.addTrip(a)
```

Appendix 11. MO Class TTVSP

Table 19 Overview and concept of proposed TTVSP MOs utilized in the AMEES for the TTVSP-HC

Name	Concept
EfficientVehicle-Rotation	Plans multiple trips of the upstream and downstream route of a line and assigns them to a new vehicle rotation
VehicleReducer	Tries to allocate every planned service trip to a reduced fleet size
VehicleTrip-Smoother	Allocates trips for individual vehicles rotations to increase the number of clock-faced headways

Listing 31 Efficient-Vehiclerotation

```
C* = ∅
for c in C:
 if h_c == ∅
  C*.addElem(c)
c = randomFrom(C*)
a* = last(AP_c)
e* = createVehicle()
while(TRUE)
 π_{a*} = 1
 e*.addTrip(a*)
 arr = a*.arrivalTime
 r_l^{u*} = R'(a*)
 for a in A_s^v
  dep = a.departureTime
  if dep == arr
   r_l^u = R'(a)
   if r_l^u == r̄_l^{u*}
    a* = a
    break
 if a* == last(e*.getTrips())
  break
add(e*)
```

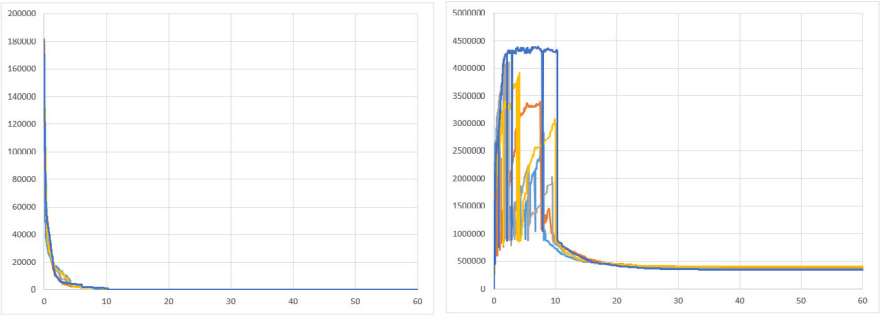
Listing 32 VehicleTripSmoother

```
for e in E
 A = e.getTrips()
 for i = 0 to numElem(A) - 1
  arr = A[i].arrivalTime
  dep = A[i + 1].departureTime
  h = dep - arr
  if h not in H^{clock}
   h* = nearest(h, H^{clock})
   π_{A[i+1]} = 0
   e.removeTrip(A[i + 1])
   a* = next(A[i + 1], h*)
   if fitsInto(a*, e)
    π_{a*} = 1
    e.addTrip(a*)
   else
    π_{A[i+1]} = 2
    e.addTrip(A[i + 1])
```

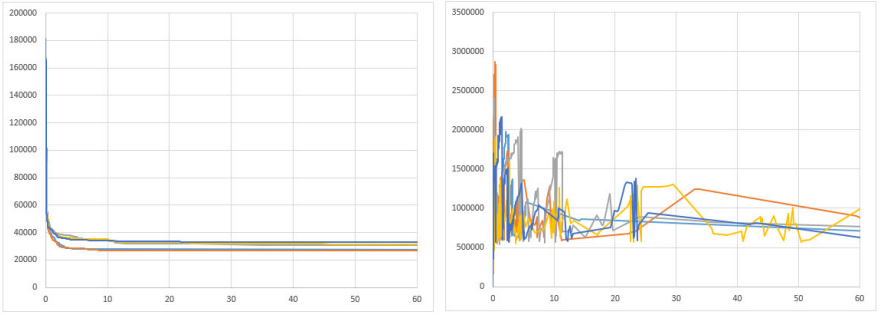
Listing 33 VehicleReducer

```
E* = ∅
for i=1 to numElem(E)-1
 e' = createVehicle()
 E*.add(e')
Δ^min = ∞
e* = ∅
for a in scheduled(A_s^v)
 for e in E*
  cost=e.getCost()
  if fitsInto(a,e)
   e.addTrip(a)
   Δ=e.getCost() - cost
   e.removeTrip(a)
   if Δ < Δ^min
    Δ^min = Δ
    e* = e
 if e* == ∅
  exit MO
 e*.addTrip(a)
for e in E
 remove(e)
for e* in E*
 add(e*)
```

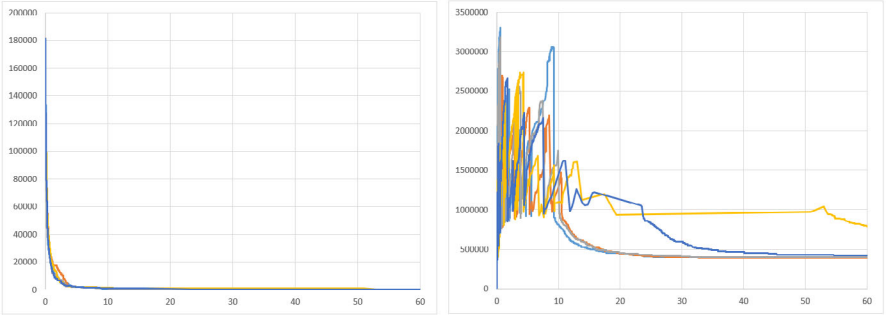
Appendix 12. Convergence of the AMEES



(a) All MOs

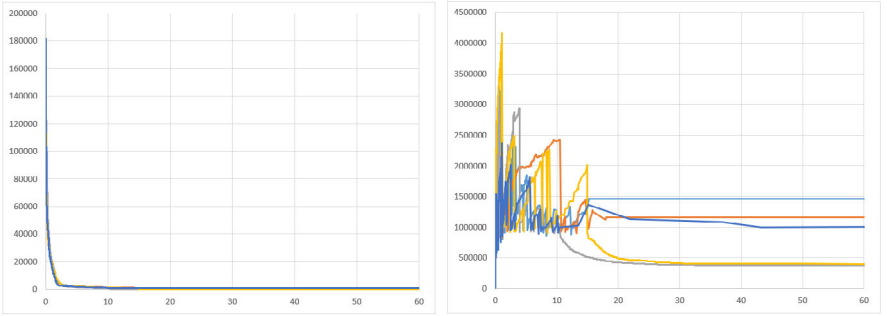


(b) No T-Planner class

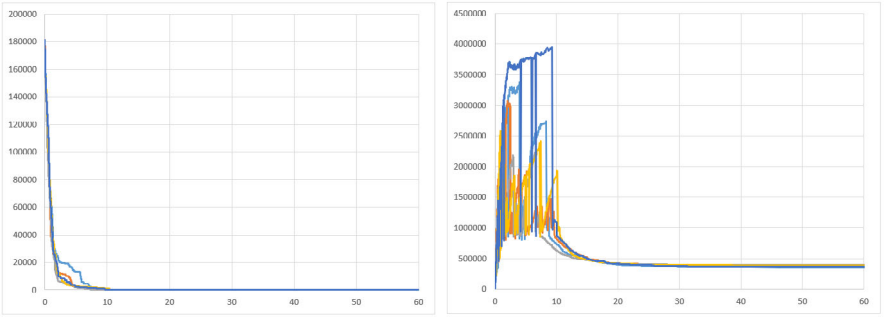


(c) No T-Unplanner class

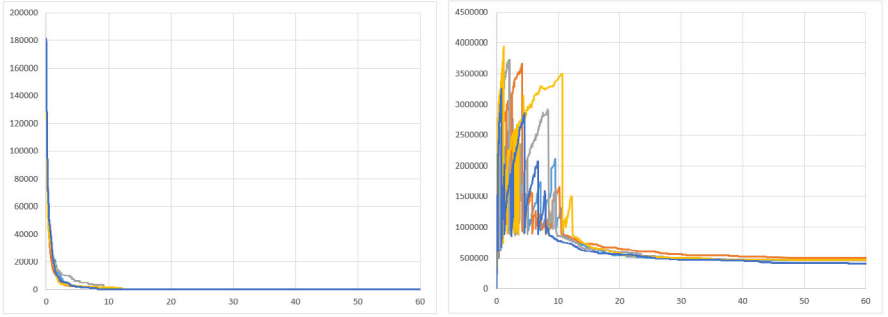
Fig. 6 Convergence of five independent runs solving the TTVSP (I4) with the AMEES. Left: invalidity (y-axis) / runtime (x-axis). Right: total cost (y-axis) / runtime (x-axis)



(d) No T-Switcher class

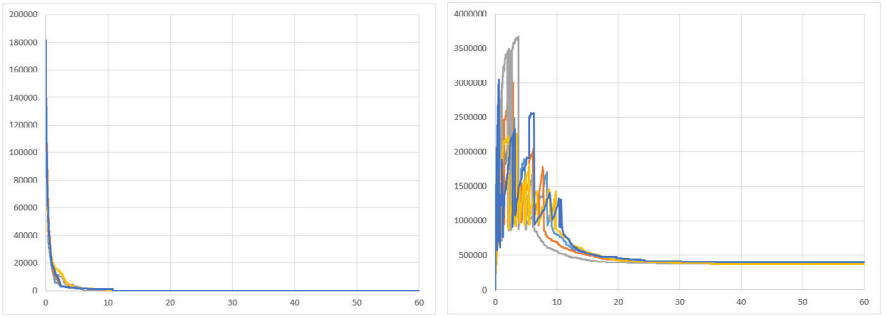


(e) No T-Allocator class

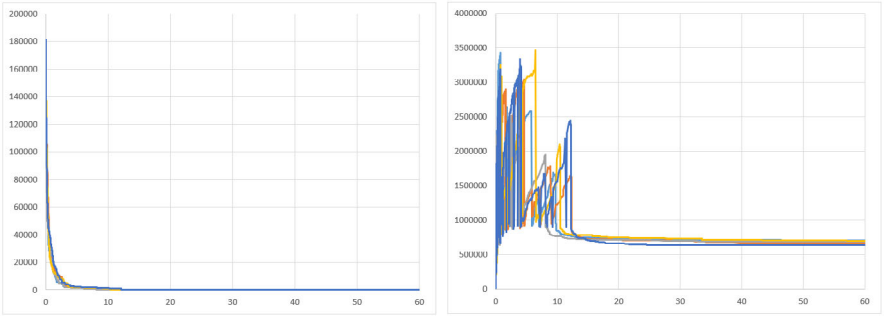


(f) No V-Merger class

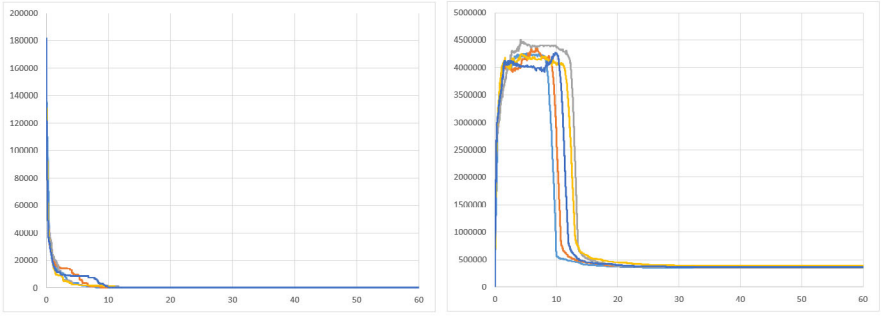
Fig. 6 continued



(g) No V-Splitter class



(h) No V-Allocator class



(i) No TTVSP class

Fig. 6 continued

Acknowledgements The authors thank the editor for the support and the anonymous reviewers for valuable feedback. Special appreciation is extended to one reviewer for their repeatedly conscientious examination and helpful suggestions in refining the manuscript.

Author Contributions L.M., N.K., and B.A. conceptualized the study. L.M. coordinated the main manuscript text and prepared the figures and tables. N.K. evaluated the manuscript repeatedly and contributed structural as well as stylistic improvements. B.A. reviewed the conducted experiments in detail, reviewed the manuscript extensively and repeatedly, and contributed improvements to all sections.

Funding Open Access funding enabled and organized by Projekt DEAL. Parts of this research have been supported by INIT Mobility Software Solutions GmbH and Inola GmbH. We gratefully acknowledge the support, particularly for the provided input data and access to the Advanced Optimization Core.

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Code Availability This paper provides pseudo-code to describe the principles of the algorithm. MIP implementations can be shared on reasonable request.

Declarations

Ethics Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

Competing Interests The authors declare no competing interests.

References

1. Desaulniers G, Hickman MD (2007) Public transit. Handbooks Oper Res Management Sci 14:69–127
2. Ibarra-Rojas OJ, Delgado F, Giesen R, Muñoz JC (2015) Planning, operation, and control of bus transport systems: a literature review. Transp Res B Methodol 77:38–75
3. Ibarra-Rojas OJ, Rios-Solis YA (2012) Synchronization of bus timetabling. Transp Res B Methodol 46(5):599–614
4. Serafini P, Ukovich W (1989) A mathematical model for periodic scheduling problems. SIAM J Discret Math 2(4):550–581
5. Ceder A (2007) Public transit planning and operation: theory, modeling and practice. Elsevier, Oxford
6. Ceder AA, Hassold S, Dano B (2013) Approaching even-load and even-headway transit timetables using different bus sizes. Public Transport 5(3):193–217
7. Klemt WD, Stemme W (1988) Schedule synchronization for public transit networks. Computer-aided transit scheduling. Springer, Berlin, Heidelberg, pp 327–335
8. Bookbinder JH, Désilets A (1992) Transfer optimization in a transit network. Transp Sci 26(2):106–118
9. Daduna JR, Voß S (1995) Practical experiences in schedule synchronization. Computer-aided transit scheduling. Springer, Berlin, Heidelberg, pp 39–55

10. Ceder A, Tal O (2001) Designing synchronization into bus timetables. Transp Res Rec 1760(1):28–33
11. Ibarra-Rojas OJ, López-Irarragorri F, Rios-Solis YA (2016) Multiperiod bus timetabling. Transp Sci 50(3):805–822
12. Kwan CM, Chang CS (2008) Timetable synchronization of mass rapid transit system using multiobjective evolutionary approach. IEEE Trans Syst Man Cybern Part C Appl Rev 38(5):636–648
13. Hassold S, Ceder A (2012) Multiobjective approach to creating bus timetables with multiple vehicle types. Transp Res Rec 2276(1):56–62
14. Gkiotsalitis K, Alesiani F (2019) Robust timetable optimization for bus lines subject to resource and regulatory constraints. Transp Res E: Logist Transp Rev 128:30–51
15. Bodin L (1983) Routing and scheduling of vehicles and crews, the state of the art. Comput Oper Res 10(2):63–211
16. Saha JL (1970) An algorithm for bus scheduling problems. J Oper Res Soc 21(4):463–474
17. Bodin LD, Rosenfield D (1976) Estimation of the operating cost of mass transit systems (No. WAHCUPS-UMTA-1-76 Final Rpt.)
18. Bertossi AA, Carraresi P, Gallo G (1987) On some matching problems arising in vehicle scheduling models. Networks 17(3):271–281
19. Bunte S, Kliewer N (2009) An overview on vehicle scheduling models. Public Transport 1(4):299–317
20. Ceder A (2001) Efficient timetabling and vehicle scheduling for public transport. Computer-aided scheduling of public transport. Springer, Berlin, Heidelberg, pp 37–52
21. Chakroborty P, Deb K, Sharma RK (2001) Optimal fleet size distribution and scheduling of transit systems using genetic algorithms. Transp Plan Technol 24(3):209–225
22. Schiewe P, Schöbel A (2022) Integrated optimization of sequential processes: general analysis and application to public transport. EURO J Transp Logist 11:100073
23. Kuo YH, Leung JM, Yan Y (2023) Public transport for smart cities: recent innovations and future challenges. Eur J Oper Res 306(3):1001–1026
24. Schiewe P (2020) Integrating timetabling and vehicle scheduling. Integrated optimization in public transport planning, vol 160. Springer optimization and its applications. Springer, Cham
25. Fonseca JP, van der Hurk E, Roberti R, Larsen A (2018) A matheuristic for transfer synchronization through integrated timetabling and vehicle scheduling. Transp Res B Methodol 109:128–149
26. Carosi S, Frangioni A, Galli L, Girardi L, Vallese G (2019) A matheuristic for integrated timetabling and vehicle scheduling. Transp Res B Methodol 127:99–124
27. Liu ZG, Shen JS (2007) Regional bus operation bi-level programming model integrating timetabling and vehicle scheduling. Syst Eng Theory Pract 27(11):135–141
28. Ibarra-Rojas OJ, Giesen R, Rios-Solis YA (2014) An integrated approach for timetabling and vehicle scheduling problems to analyze the trade-off between level of service and operating costs of transit networks. Transp Res B Methodol 70:35–46
29. Liu T, Ceder AA (2017) Integrated public transport timetable synchronization and vehicle scheduling with demand assignment: a bi-objective bi-level model using deficit function approach. Transp Res Procedia 23:341–361
30. Liu T, Ceder A, Chowdhury S (2017) Integrated public transport timetable synchronization with vehicle scheduling. Transp A: Transp Sci 13(10):932–954
31. Kliewer N, Mellouli T, Suhl L (2006) A time–space network based exact optimization model for multi-depot bus scheduling. Eur J Oper Res 175(3):1616–1627