Transportation Research Part B 127 (2019) 99–124

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

A matheuristic for integrated timetabling and vehicle scheduling

Samuela Carosi[a], Antonio Frangioni[b], Laura Galli[b,*], Leopoldo Girardi[a], Giuliano Vallese[a]

[a] OR division, M.A.I.O.R. Srl, Lucca, Italy
[b] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, Pisa, 56127, Italy

ARTICLE INFO

Article history:
Received 20 July 2018
Revised 4 July 2019
Accepted 13 July 2019

Keywords:
Public transport
Timetabling
Vehicle-scheduling
Integrated approach
Matheuristic

ABSTRACT

Planning a public transportation system is a complex process, which is usually broken down in several phases, performed in sequence. Most often, the trips required to cover a service with the desired frequency (headway) are decided early on, while the vehicles needed to cover these trips are determined at a later stage. This potentially leads to requiring a larger number of vehicles (and, therefore, drivers) that would be possible if the two decisions were performed simultaneously. We propose a multicommodity-flow type model for integrated timetabling and vehicle scheduling. Since the model is large-scale and cannot be solved by off-the-shelf tools with the efficiency required by planners, we propose a diving-type matheuristic approach for the problem. We report on the efficiency and effectiveness of two variants of the proposed approach, differing on how the continuous relaxation of the problem is solved, to tackle real-world instances of bus transport planning problem originating from customers of *M.A.I.O.R.*, a leading company providing services and advanced decision-support systems to public transport authorities and operators. The results show that the approach can be used to aid even experienced planners in either obtaining better solutions, or obtaining them faster and with less effort, or both.

1. Introduction

Public transportation companies often face complex logistic problems. In particular, vehicles and crews represent expensive resources for the operators, that require efficient utilization. Planning in a public transportation system is usually decomposed into stages, that are solved in sequence, namely: Network Design (ND), Line Planning (LP), TimeTabling (TT), Vehicle Scheduling (VS) and Crew Scheduling (CS). The first three steps define the type of service to be offered: ND and LP determine the set of lines (and connections) and how often the service is offered along the lines, while TT defines the departure and arrival time of the individual trips on each line in order to meet the desired frequency of service. The last two steps are, instead, devoted to resource allocation: VS is the assignment of buses to trips, such that each trip is covered by exactly one bus and the schedules of all the vehicles are feasible, while CS is the assignment of crews to trips, such that each trip is covered by a crew and all the crew schedules satisfy the required logical and legal restrictions. We

* Corresponding author.
E-mail addresses: samuela.carosi@maior.it (S. Carosi), antonio.frangioni@unipi.it (A. Frangioni), laura.galli@unipi.it (L. Galli), leopoldo.girardi@maior.it (L. Girardi), giuliano.vallese@maior.it (G. Vallese).

https://doi.org/10.1016/j.trb.2019.07.004
0191-2615/

refer the reader to Desaulniers and Hickman (2007) for a detailed description of the various stages, and to Guihaire and Hao (2008) for a global review of the crucial strategic and tactical steps of transit planning.

There is a vast literature addressing each one of the above steps individually. Yet, because of the interdependence of the stages, planning in sequence possibly produces suboptimal solutions. This is in particular true for the vehicles and drivers needed, that are only determined in the later steps of the planning process. Unfortunately, decomposing into stages is often necessary to make the solution time compatible with the requirements of the planners. The two intermediate stages TT and VS are generally argued Ceder and Wilson (1986) to be "the bulk" of the decision process. Indeed, many recent developments in transit planning, including this work, focus on the *integration* of these two steps.

In particular, our contribution consists in a new model for the Integrated Timetabling and Vehicle Scheduling (ITTVS) problem. Under some assumptions on the VS constraints, that are particularly reasonable for the urban planning context and can be somewhat relaxed, the model is a compact multicommodity-flow type problem; however, its size and the relative weakness of the continuous relaxation are such that the problem cannot be solved by off-the-shelf tools with the efficiency required by the planners. We therefore also propose a diving-type matheuristic approach for the problem, which produces good-quality solutions in reduced time. We report experiments on several real-world ITTVS instances originating from customers of *M.A.I.O.R.*, a leading company providing services and advanced decision support systems to public transport authorities and operators, showing that good-quality solutions—in particular, if compared with those manually constructed by experts of the transport companies and currently used in operations—can be obtained with a reasonable computational effort. The variant of the approach where the continuous relaxation of the model is tackled by forming its Lagrangian relaxation w.r.t. the linking constraints, and approximately solving the corresponding Lagrangian Dual by means of a bundle-type method, appears to be particularly promising as the size and complexity of the instances grow.

The structure of the paper is as follows. Section 2 contains the literature review. Section 3 presents the base case scenario for our real-world application, which is mathematically formulated in Section 4. Section 5 discusses some important extensions to the base case scenario. Our *matheuristic* approach is described in Section 6, and computational results are discussed in Section 7. Finally, in Section 8 we draw some conclusions.

2. Literature review

In this section, we first provide a general description of the TT and VS as individual steps. We then review the literature dealing with attempts at integrating the two phases, providing a taxonomy that allows us to frame our contribution. We also briefly comment on the relationships between the problem we face and related ones, e.g., for different kind of transportation systems.

2.1. Timetabling

Timetabling (TT) is the process of creating a schedule starting from the route network and the desired frequency of service. The result is a set of trips, with the scheduled times at the terminals and major points on the routes, a.k.a. the *timetable*. Timetabling can be *periodic* ("clock-face") or *non-periodic*. If the order of the events is fixed, the latter can be efficiently solved by shortest path techniques. If events appear periodically, an ordering is not possible, this is why the *periodic event scheduling problem* (PESP) is $\mathcal{NP}$-hard Serafini and Ukovich (1989). In non-periodic TT one usually measures the *headway* of a line, i.e., the time separating the service at its main stop by consecutive runs; this specifies how often bus service should be offered, and is basically the inverse of the *frequency* usually considered in periodic timetabling.

The TT problem aims at finding "good quality" timetables from the viewpoint of users of the transportation service. This may mean different things. Perhaps the simplest one is *regularity*, whereby one seeks to find a timetable where the trips have exactly the frequency/headway required for the line they belong to (in the corresponding time window), or at least the distance of the actual frequency from the desired one is minimized. This is the only reasonable measure if the topology consists of a single—albeit, possibly, "complex"—line, as in our experiments. However, when multiple lines are considered, the *transfer coordination* or *synchronization* variant is frequently studied, where one is rather interested in finding schedules that minimize transfer and/or waiting time of passengers (or other synchronization measures) at the stops connecting different lines. That is, the aim is to coordinate the trips on different lines; clearly, this requires modelling passengers' waiting and transfer activities during vehicle changes.

In the context of transit planning, TT is included within *operational* planning. The reason is twofold; (*i*) timetabling occurs frequently (e.g., every 3–6 months); (*ii*) it is from the timetables that vehicle and crew schedules are constructed. Yet, the goal of timetabling is a *tactical* one, since it aims at *optimizing passengers' service*. This is in contrast to the VS and CS, that are typically intended to *minimize operating costs*.

2.2. Vehicle scheduling

If the lines and the timetable are given, so is the set of *trips*, i.e., sequences of arrival/departure times at each stop of a given line, that must be operated by the same vehicle. The set of trips is the input of VS, which aims at optimally covering them, typically minimizing the number of vehicles needed and/or some other measure of the required effort, such as deadheads (i.e., vehicle movements that do not constitute transportation service) or other operating costs, while meeting all operational constraints. VS plays an important role in the public transport planning process, since it is the first planning step where the primary focus is put on *minimizing* costs, while previous steps typically focus on passenger service. The vehicle scheduling problem is the task of building an optimal set of sequences of trips, each sequence—called *vehicle schedule*—to be performed by an individual vehicle, such that each trip of a given timetable is covered by exactly one sequence. A sequence of trips assigned to a vehicle results in a *vehicle route*, that can serve several lines (*interlining*). *Multi-depot* VS is $\mathcal{NP}$-hard Carpaneto et al. (1989), while the *single-depot* case can be solved in polynomial time Bertossi et al. (1987), provided there are no constraints on how a chain can be formed, apart from compatibility between two trips (taking into account max/min waiting time at terminals and/or deadheading, if allowed). More complex variants consider different types of vehicles (e.g., number of seats, level of comfort, etc.). A well-known modelling framework for the VS problem is based on a *time-space network* Bertossi et al. (1987), where vertices are departures or arrivals of a vehicle at a specific time and location, such as the beginning or end of a service trip, and edges link two actions that can be performed by the same vehicle. A vehicle schedule corresponds to a flow through the network, so that the computation of the optimal vehicle schedule can be performed by calculating a minimal cost circulation through the network, with additional constraints guaranteeing that all service trips are performed exactly once.

2.3. Integrated TT and VS

With only one exception, all the works in the literature considering integrated timetabling and vehicle scheduling in urban public transport deal with the *transfer coordination* version of the TT, i.e., where the objective is to minimize the transfer and waiting time for passengers. To the best of our knowledge, the first two papers are Ceder (2001) and Chakroborty et al. (2001). The former presents a 4-step sequential approach with a single feedback loop that determines a timetable and the corresponding vehicle schedules. The solution approach of the latter, instead, is based on a genetic algorithm to simultaneously optimize the fleet size without interlining (i.e., each bus can serve only one line) and the waiting and transfer time of passengers.

In general, a crucial characteristic of all approaches is that the integrated problem has a *bi-objective* nature; that is, it aims simultaneously at maximizing the *timetable quality* from the passengers' point of view, and minimizing the *operating cost* of vehicle schedules from the service provider's point of view. Clearly, these two objectives are potentially in contrast to each other; thus, a main decision, when developing an integrated model is on how the interaction between the two contrasting objective functions should be managed. Correspondingly, we subdivide all the articles in the literature according to the strategy they adopt in this respect:

- *Shifting*. An important stream of research is based on the idea of solving the VS problem allowing some flexibility to change the timetable, thus leading to the *Vehicle Scheduling with Time-Windows* (VSP-TW) problem. That is, the timetable is given as an input, and arrival times can only be modified (shifted) by a small amount, in order to allow for cheaper vehicle schedules. Clearly, this approach prioritises the service provider's objective function (operating cost); however, the quality of the timetable is somewhat guaranteed by the fact that only minor modifications, w.r.t. the nominal one, are allowed. Hence, in bi-objective parlance these methods are akin to *budgeting* ones, where one objective is optimized subject to the constraint that the other one cannot become worse than a given threshold (although in this case the threshold is only indirectly specified).
- *Weighting*. This approach consists in having, as objective function, the *weighted* sum of the two original ones. The issue with this kind of approaches is typically that of finding weights that accurately represent the preferences of the decision maker.
- *Pareto-front*. To account for the inherent difficulty of the two previous approaches, i.e., that of selecting either an appropriate budget or appropriate weights, it is possible to try to produce a set of Pareto-optimal (i.e., non dominated) solutions. This can be done, for example, by solving the budgeted/weighted versions of the problem with several choices of the budget/weights; alternatively, *population-based algorithms* can be used, as they naturally generate multiple solutions.
- *Bilevel programming*. This approach takes a different stance, where the *leader* (say, the service provider) optimizes its objective function, while the *followers* (say, the users) react by optimizing their own (say, their travel time), subject to the leader choices.
- *Reordering*. Finally, in this specific context, the idea has been proposed that it might be possible to obtain "more integrated" solutions by simply reordering the classic sequence of the planning steps outlined at the beginning of this section.

We will now briefly describe all the papers in the literature as "subdivided" among the five above categories. As it often happens our taxonomy is only approximate, as some contributions combine different strategies within the same algorithmic framework (e.g., Fleurent et al., 2007; Fleurent et al., 2009; Fonseca et al., 2018). It might be appropriate to mention at this point that, due to the complexity of the problem, most of these studies propose *meta-heuristic* algorithms such as Iterated Local Search (ITL), Tabu-Search (TS), Large Neighborhood Search (LNS), Genetic Algorithms (GA), and Simulated Annealing (SA).

2.3.1. Shifting

This kind of approach can be traced back to the seminal paper Kliewer et al. (2005), which considers (small) time windows in which the departure time of a service trip can be shifted, and use a time-space network to determine feasible trip combinations. The solution approach is based on column generation together with heuristics. The model of Kliewer et al. (2005) is extended in Van den Heuvel et al. (2008) and Fleurent et al. (2007), where an hierarchical approach for VS is developed, combining mathematical programming models, to optimize the type and the number of vehicles for each trip, with a SA approach, that allows the trip starting times to be shifted in time. Similarly, the use of VSP-TW in the context of tactical timetable analysis is discussed in Bunte and Kliewer (2009), where it is suggested to model the VSP-TW as a *Vehicle Routing Problem with Time Windows* (VRP-TW) and to estimate the potential of vehicle savings for a given timetable by allowing wider departure time windows (up to 20 min) for service trips. Recently, Fonseca et al. (2018) proposes a matheuristic that combines the idea of *shifting* with that of *weighting*. The algorithm iteratively solves a bi-objective mathematical formulation (minimization of passenger transfers and operational costs) of the ITTVS allowing timetable modifications for a subset of timetabled trips, while solving the full VS problem. A similar approach, combining shifting with weighting, is used in the interactive tool `NetPlan` described in Fleurent et al. (2007, 2009), which integrates timetabling and vehicle scheduling. The tool is developed by `GIRO`, a privately owned company based in Montreal that provides software and services to plan and manage public transport operations. Finally, Daduna and Paixão (1995) provides an instructive overview of different restrictions for the VS and show practical applications in urban mass transit companies.

2.3.2. Weighting

The ITTVS with *time windows and balanced departure times* is studied in Schmid and Ehmke (2015); the problem is modeled as a VRP-TW, that includes the balancing of trips departure times and minimization of deadheads in the objective function, and it is solved by a hybrid LNS approach that decomposes the problem into a scheduling and a balancing component. The *Simultaneous Vehicle Scheduling and Passenger Service Problem* (SVSPSP) has been defined for the first time in Petersen et al. (2012), where an integrated solution approach is proposed, based on the LNS used in Desrosiers et al. (1995) to solve the multiple depot vehicle scheduling problem (MDSVP); the approach is tested on the Greater Copenhagen Area. A solution approach based on TS is presented in Guihaire and Hao (2010), where at each iteration the timetable is altered and the optimal trip assignment is recomputed solving a linear quasi-assignment problem. Finally, as already mentioned, both Fonseca et al. (2018) and Fleurent et al. (2007, 2009), integrate timetabling and vehicle scheduling using a weighted objective function.

2.3.3. Pareto-front

Two integer linear programming models for TT and VS are defined in Ibarra-Rojas et al. (2014) and combined in a bi-objective integrated model that is solved repeatedly using a *budgeting* approach. A ITTVS model (without interlining) is presented in Weiszer et al. (2010) and solved by the direct application of a *multi-objective* GA.

2.3.4. Bi-level

A bi-level ITTVS integer programming model is developed in Zhi-gang and Jin-sheng (2007) and it is solved using a specialized TS algorithmic framework. A more complex bi-objective and bi-level approach is presented in Liu and Ceder (2017) and Liu et al. (2017) to study how much the changes on timetable and vehicle scheduling affect users trips choice behaviour. In the model, the upper level is a service provider, that creates timetables and vehicle schedules to minimize total operating costs and passenger waiting/travel time, while the lower level are public transport users, who choose their travel paths in a user optimal manner, responding to the operator decision (transit assignment problem).

2.3.5. Reordering

A “reverse shifting” approach is proposed in Guihaire and Hao (2007) and tested on real-life instances from France; the input is the current timetable, vehicle and crew schedule, and the timetable is adjusted by a TS approach keeping the vehicle and driver schedules fixed. In Michaelis and Schöbel (2009) the process starts by designing the vehicle routes; then these routes are interpreted as lines and the corresponding frequency is defined, finally the timetabling phase assigns an arrival and a departure time to each stop of the route. The objective function is designed in order to measure the “attractiveness” for passengers, using an origin-destination matrix of potential transport demands and maximizing the probability that a (potential) traveler between two locations decides to use public transportation rather than a private one. The heuristic is applied to a case study that optimizes the local bus system in Gottingen. More recently, Pätzold et al. (2017) considers three consecutive planning stages in public transportation (LP, TT and VS) and propose three different ways to “look ahead”, i.e., to include aspects of vehicle scheduling already earlier in the sequential process, including a reordering of the sequential planning stages.

2.4. Related problems

Similar issues as ITTVS also appear in the management of different transportation systems, like railways. In that setting, VS corresponds to *rolling stock* scheduling. The problems are rather different in nature, since a train can be “(de)composed”, whereas a bus can only consists of one indivisible unit. Yet, models based on time-space graphs have been widely used in both settings Schrijver (1993). For railway transport in particular, Borndörfer et al. (2015) and Rangaraj et al. (2006) address the topic of integration for rolling stock planning.

With regards to periodic timetables, we mention some recent research works that provide interesting insights into the interplay between timetabling and vehicle scheduling, but without actually integrating them.

The software toolbox LinTim Schoebel et al. (2018) gives the possibility to solve the various planning steps in public transportation, including line planning, periodic timetabling, delay management and rolling stock circulation. The integration of different steps into a common environment allows one to analyze the mutual influence of the planning steps.

The recent Borndörfer et al. (2018) shows that the number of vehicles that are required to operate a given periodic timetable can be computed efficiently by solving a perfect matching problem, rather than using the common time-expanded network flow model approach.

Finally, in Borndörfer and Liebchen (2008) the authors compare *periodic vs aperiodic* timetabling with respect to vehicle operation costs.

2.5. Contributions of this paper

As already mentioned, almost all the previously cited articles focus on the *transfer coordination* version of the TT, save for Schmid and Ehmke (2015), where *regularity* (i.e., minimizing the deviation from the desired headways) is considered. Also, almost all the contributions use meta-heuristics, save for Fonseca et al. (2018), where a matheuristic approach is developed for an integrated bi-objective formulation (but with the transfer coordination objective function).

The ITTVS problem at *MAIOR*, described in detail in Section 3, is characterized by the following features: (i) A non-periodic timetabling problem, with regularity (i.e., deviation from the ideal service frequency) objective function; (ii) a single depot, single vehicle type vehicle scheduling problem. The contributions of the paper are the following. First, we consider a real-world bus planning application at *MAIOR* and present an integrated solution approach for two steps (i.e., TT and VS), that were previously solved in sequence by customers of the company. This allows us to test our integrated approach on real-world instances provided by Italian public transport providers, comparing them with those previously produced by the sequential approach. Thus, we are able not only to compare the objective values, showing that the integrated approach significantly reduces them, but also to have our solutions evaluated by expert transport planners, which certified them to be of “good quality” according the their judgement. This is important, because our integrated approach is based on weighting, and the proper selection of weights is crucial for the practical quality of the solution. Moreover, to our knowledge, our approach is the first matheuristic for ITTVS with regularity objective function (minimizing deviation from the ideal frequency of service). Finally, we consider some extensions for dealing with “complex” single-line topologies and constraints on the number of vehicles, which are important for the practical applicability of the approach in some customers’ environments.

3. Problem description

This section describes the specific characteristics of the “base case” ITTVS problem at *MAIOR*, where the topology is that of a *simple single line*. This is only for simplicity of exposition, in that the mathematical formulation presented in Section 4 immediately extends to multiple independent lines (that is, independent from the TT side, while potentially linked in the VS one). Less trivial extensions are shown in Section 5, in particular for when multiple routes (besides the two obvious ones) actually pertain to the same line (i.e., a *complex single line*).

The main input to the integrated TT-VS problem is a *public transportation network* (PTN), a set of *potential* trips $\mathcal{T}$ (i.e., not every trip has to be operated by some vehicle), and the desired (a.k.a. *ideal*) headways for each of the *time windows* in which the operating interval is subdivided, and for each direction. In general, a PTN is given in the form of a graph, where the nodes correspond to bus stops or depots, and the links correspond to direct bus transits; however, the actual graph description of the PTN is inconsequential for our treatment. A *simple (single) line* is a bi-directional path AB in the PTN between two *terminals* A and B (i.e., start/end stops of a line). Usually a simple line has two directions, called *in-bound* and *out-bound* and denoted by $\mathcal{D} = \{\vec{AB}\ \vec{BA}\}$, respectively; however, a *circular* line (where $A = B$) may have only one direction. In more general cases, a single line may comprise multiple routes or *patterns* for each direction, as discussed in Section 5; however, in this paragraph patterns and directions coincide, as shown in Fig. 1. A *trip* corresponds to a pattern/direction in the PTN that has to be operated by some vehicle at a given time. Since each trip belongs to a given pattern/direction, we define $\mathcal{T} = [\mathcal{T}_d]_{d\in\mathcal{D}}$ as the “direction partition” of $\mathcal{T}$. Each trip $i \in \mathcal{T}$ is characterized by a start and end terminal, denoted by $sn(i)$ and $en(i)$, respectively, while the corresponding departure time from $sn(i)$ and arrival time at $en(i)$, are denoted by $st(i)$ and $et(i)$, respectively. Note that all the potential trips in $\mathcal{T}$ have to be separately described in input, as the trip times along the line can vary considerably during the planning horizon (e.g., at “peak times” headways are much shorter), and also within a single time window and for different directions. Thus, it is not sufficient to just describe a single travel time to characterize $\mathcal{T}$, as it often happens in the timetabling literature.

Even in the case of a simple single line, for VS purposes it is necessary to consider in the PTN, besides the terminal nodes A and B, also the *single depot* node O (but not any other intermediate stop of the line).

In the following, we will denote by N the set of terminals of the involved lines (say, $N = \{A\ B\}$) and by $N^+ = N \cup \{O\}$. For each direction, a *main stop* is identified, symbolized by a “clock” in Fig. 1, which is used to calculate the *headways*. Although the figure may suggest that the main stop needs be the same for the two directions of a simple line, this is not necessarily

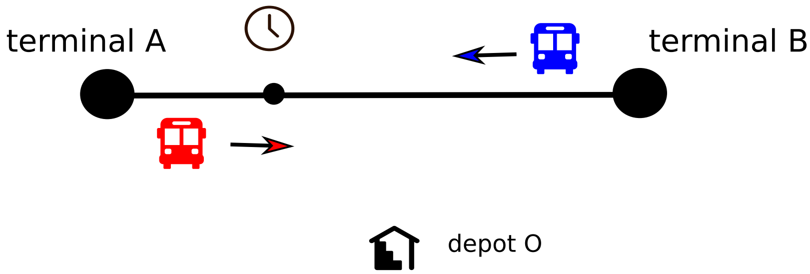

Fig. 1. Simple single line.

true in practice (especially since the stops along the two directions could be disjoint). The choice of the main stop can vary, depending on the structure of the line. Usually it is a "busy" point of the line, with high passenger demand, for which the planners are interested to monitor service frequency. It may coincide with one of the terminals if it is a relevant location of the line (e.g., a railway node).

3.1. TT components

Each trip $i \in \mathcal{T}$ is associated to a uniquely identified pattern/direction $d(i)$. The trip specifies the arrival time at each stop of the pattern, including the arrival time $a(i)$ at the main stop of $d(i)$. Since we assume only one vehicle type, the arrival times of a given trip are the same for all the vehicles. A *timetable π_d for a direction $d \in \mathcal{D}$* is a subset of its potential input trips $\mathcal{T}_d$; a timetable is then just the union of $|\mathcal{D}|$ (independent) timetables, one for each direction, i.e., $\pi = [\pi_d]_{d \in \mathcal{D}}$. Given a timetable π, the (actual) headways of a direction d w.r.t. π are the times separating each two consecutive trips i, j in π_d passing at its main stop, i.e., $a(j) - a(i)$. In our non-periodic planning, a time horizon T is given; say 5:00–24:00, i.e., each day is treated independently. As the desired frequency of service typically varies along the day, T is partitioned into k *time windows* defined by $k+1$ time instants $t_0, t_1, \dots, t_k$, where t_0 and t_k are the initial and final time instants of T. For each time window h and each direction $d \in \mathcal{D}$, we are given the *ideal headway* I_d^h, together with *minimum and maximum headways* $\underline{I}_d^h \le I_d^h \le \bar{I}_d^h$. We will denote by $h(i)$ the time window in which trip i happens; for simplicity we will only describe the case in which trips are completely contained in a time window, and therefore the ideal, minimum and maximum headways of the trip are simply defined as $I_i = I_{d(i)}^{h(i)}$, $\underline{I}_i = \underline{I}_{d(i)}^{h(i)}$, and $\bar{I}_i = \bar{I}_{d(i)}^{h(i)}$, respectively. However, only minor changes are required to account for "border effects" when trip i falls in two consecutive time windows, say h and $h+1$. For instance, one can take $\underline{I}_i = \min\{\underline{I}_{d(i)}^h\ \underline{I}_{d(i)}^{h+1}\}$ and $\bar{I}_i = \max\{\bar{I}_{d(i)}^h\ \bar{I}_{d(i)}^{h+1}\}$; as for I_i, one can take a convex combination of $I_{d(i)}^h$ and $I_{d(i)}^{h+1}$ whose weights can be chosen in different ways (such as how much of the trip falls in each time window).

With the above definitions, a *feasible timetable* $\pi_d \subset \mathcal{T}_d$ for a direction $d \in \mathcal{D}$ is a timetable such that:

- The (actual) headway of each two consecutive trips i and j in π_d is feasible, i.e., $a(j) - a(i) \in [\underline{I}_i\ \bar{I}_i]$;
- The *first* and the *last trip* of π_d belong to given subsets $\mathcal{T}_d^{ini}$ and $\mathcal{T}_d^{fin}$ of *initial* and *final* trips, specified as an input to the problem.

To evaluate the quality of a timetable, a quadratic *penalty function* is given specifying how to compute the cost of the deviation of a feasible actual headway $\bar{a} \in [\underline{I}_d^h\ \bar{I}_d^h]$ from the ideal one I_d^h. The actual form of this function is immaterial for our model, just assuming the trivial properties that the penalty is zero if $\bar{a} = I_d^h$, and larger than zero (typically, nondecreasing in $|\bar{a} - I_d^h|$) otherwise.

3.2. VS component

In the VS literature, traveling between two trips without passengers on board is called a *deadhead trip*. In particular, a vehicle leaving a depot to reach the start-terminal of a trip is said to be performing a *pull-out trip*; similarly, it performs a *pull-in trip* when it returns to the depot from the end-terminal of a trip. For each node $n \in N^+$ and for each time window h we are given a *minimum* and a *maximum stopping-time*, denoted by $\underline{\delta}_n^h$ and $\bar{\delta}_n^h$, respectively; however, we typically assume that there is no maximum stopping time at the depot, i.e., $\bar{\delta}_O^h = +\infty$ for all h. Note that we do not consider stopping times for any intermediate node of the given line. For each terminal $n \in N$ and for each time window h, we are also given the *travel time* for a pull-in and pull-out trip, denoted by t_{n+}^h and t_{n-}^h, respectively. In general, two trips are said to be *compatible* if they can be covered consecutively by the same vehicle. In our application, we distinguish two types of compatibilities between two trips $i, j \in \mathcal{T}$:

- *In-line compatibility* means that $en(i) = sn(j)$, i.e., trip j starts at the same terminal in which i ends, and $\underline{\delta}_{en(i)}^{h(i)} \le st(j) - et(i) \le \bar{\delta}_{en(i)}^{h(i)}$, i.e., the waiting time at the terminal between the end of trip i and the start of trip j is feasible;

- *Out-line compatibility* means that $en(i) \neq sn(j)$ and $st(j) - et(i) \geq t^{h(i)}_{en(i)+} + \underline{\delta}^{h(i)}_{O} + t^{h(j)}_{sn(j)-}$; in other words, there must be enough time between the end of trip i and the start of trip j to perform a pull-in trip from $en(i)$, wait the minimum amount of time at the depot, and then perform a pull-out trip towards $sn(j)$. Note that pull-in and pull-out trips are not included in $\mathcal{T}$, as they are not (passenger) service trips (i.e., no passengers on board).

In our case study, if $en(i) \neq sn(j)$, the vehicle cannot move directly from one terminal to the other, but it must perform an out-line compatibility. In other words, we only allow deadhead trips that start or end at the depot (i.e., pull-in/pull-out trips). Yet, it could make sense to have deadheading without touching the depot, subject to time compatibility. This can be easily accounted for, without impacting the general structure of our model (barring some specific details discussed later on). A *feasible schedule for a vehicle* is composed of an initial pull-out trip, a sequence of compatible (service) trips in $\mathcal{T}$, possibly separated by pull-in/out trips, and a final pull-in trip to return to the depot. In general, feasible schedules for a vehicle can be seen as sequences of *vehicle blocks*, where each block consists of a sequence of (service) trips in $\mathcal{T}$, that starts and ends at the depot without returning to it in the middle of the sequence. A feasible *vehicle schedule* Ω is a subset of the input potential trips $\mathcal{T}$ that can be partitioned in feasible schedules for single vehicles, possibly satisfying a constraint on the maximum number of vehicles to be used if it is imposed.

3.3. Integrated problem

The objective of our integrated problem is to provide a solution that optimally balances the service provider cost (VS objective) and the users satisfaction (TT objective). The latter is simply captured by minimizing the sum of the costs of all the deviations between the actual headways and the desired ones, each one measured by the penalty function alluded to above. The former is somewhat more complex. Since one of the main costs for the service provider is usually the number of vehicles used, the primary VS objective is the minimization of the number of bus schedules. Two secondary measures of the service provider cost are the time spent by the vehicles waiting at the terminals in excess to the minimum waiting time (for drivers will typically have to man them even when stationary, thus increasing labour cost), and the time spent by the vehicles performing pull-in and pull-out trips (for the same reason as above, plus the fact that vehicles typically consume some fuel). The relative importance of these terms is defined by weighting parameters in the VS objective function; this is in addition to the weights given to the two different overall objective functions (TT and VS) (as described in Section 2.3.2), but the selection of the sub-weights for the VS part is typically done even when solving the problem by separate phases, and therefore these are well-established in practice (also because they can ultimately be reduced to monetary costs). The selection of the weights for the primary objective functions is more delicate, which is why judgement by experts was required to evaluate the solutions produced by the integrated approach before the results could be deemed satisfactory. Note that, on the other hand, having both objective functions can actually help in properly modelling some aspects of the transportation system. For instance, nowhere in the VS part the capacity of the vehicles is explicitly taken into account. Indeed, this is not necessary since this aspect is taken care of in the TT subproblem, since the maximum headways are typically computed precisely to prevent overcrowded services.

4. Mathematical model

We now present the mathematical model for the ITTVS as described in the previous section. We will often make reference to the “base case” scenario of a simple single line for illustration, although the model readily extends to any number of lines (patterns). The model consists of $|\mathcal{D}|$ *TT sub-problems*, one for each direction, a single *VS sub-problem*, and some *linking* constraints, that guarantee integration. The TT sub-problems select an optimal subset of trips $\mathcal{T}^* \subset \mathcal{T}$, corresponding to feasible timetables $\pi^* = [\pi^*_d]_{d \in \mathcal{D}}$ for all directions, that minimize the total cost of deviation from the ideal headways. The VS sub-problem constructs a feasible vehicle schedule Ω^* with minimum operator cost out of the selected trips $\mathcal{T}^*$; in other words, Ω^* is a *vehicle schedule cover* of π^*. Clearly, the subproblems are not independent since the vehicle schedule depends on the choice of $\mathcal{T}^*$, which is what the linking constraints provide by ensuring that the trips covered by Ω^* correspond to all and only the trips used in the timetable.

We propose a *Mixed Integer Linear Programming (MILP)* multicommodity flow-type model, based on node-arc formulations where arc flow variables represent either the timetables or the vehicles schedule. That is, we construct one directed graph for each of the TT subproblems (direction d) and one directed graph for the VS subproblem, as described in Section 2.1 and 2.2, respectively. Finally, the integrated MILP formulation, comprising the linking constraints, is shown in Section 4.3.

4.1. TT model

The TT model is based on representing feasible timetables in terms of paths on a directed *TT graph* $G^{TT}_d = (N^{TT}_d\ A^{TT}_d)$, which is a *compatibility graph*. For a given direction $d \in \mathcal{D}$, the nodes of the corresponding TT graph represent the trips in $\mathcal{T}_d$ plus a dummy source O^-_d and a dummy sink O^+_d: $N^{TT}_d = \mathcal{T}_d \cup \{O^-_d\ O^+_d\}$. The arcs in A^{TT}_d leaving the source node O^-_d end in the nodes corresponding to the set of potential initial trips $\mathcal{T}^{ini}_d$, and symmetrically for those entering the sink node O^+_d. An arc $(i, j) \in A^{TT}_d$ between two trips $i, j \in \mathcal{T}_d$ exists if and only if i and j are neither “too close” or “too far apart”, i.e., the corresponding headway is feasible. Its cost is computed off-line with the selected penalty function, which can therefore be

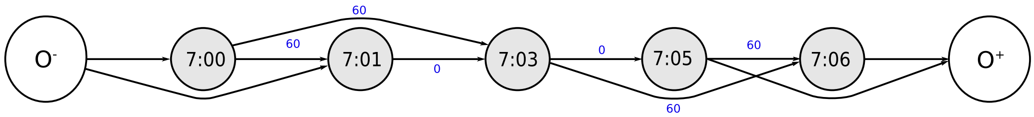

Fig. 2. G_d^{TT} compatibility graph.

arbitrary. It is trivial to see that G_d^{TT} is acyclic and that a path between O_d^- and O_d^+ in G_d^{TT} corresponds to a feasible timetable π_d, the cost of the path (sum of the costs of the arcs) being the total cost of violation of ideal headways. G_d^{TT} being acyclic, each TT sub-problem—were they independent, which they are not—could be easily solved as an *acyclic shortest path (SP)* problem, whose complexity is linear in $|A_d^{TT}|$ and therefore at worst quadratic in $|\mathcal{T}_d|$ (but in practice basically linear in $|\mathcal{T}_d|$, since many trips are not compatible due to the constraints on the minimum and maximum headway).

Example 1. Consider the small example in Fig. 2 with 5 trips. G_d^{TT} consists of two dummy nodes (*source* O_d^- and *sink* O_d^+) and 5 trip nodes, for simplicity all belonging to the same time window. The time indicated inside the trip nodes represents the arrival time $a(i)$ at the *main stop*. The ideal headway is 2 min, with minimum and maximum headway of 1 and 3 min, respectively. The cost of the arcs (in blue) is computed using as simple penalty function the absolute value of the deviation from the ideal headway, in seconds. The minimum cost feasible $O_d^- - O_d^+$ path in the graph corresponds to the timetable 7:01–7:03–7:05, that has a cost of 0, which means that the optimal total deviation from the ideal headways is 0, as it is immediate to verify. Note that we did not draw all the possible arcs just for the sake of simplicity.

4.2. VS model

The VS model is based on representing feasible vehicle schedules in terms of flows on a *single* directed *VS graph* $G^{VS} = (N^{VS}\ A^{VS})$, which is also basically a compatibility graph: The VS sub-problem is not separable per direction, because the schedule for each single vehicle can–and usually does—cover trips belonging to different directions. Using compatibility graphs to represent VS is well-known in the literature Bertossi et al. (1987); however, in a standard VS, such as when the problem is solved in the classical planning sequence, one typically has to construct feasible vehicle schedules that cover all the trips of some *input* timetable π^*. In our ITTVS, instead, the optimal timetable π^* is unknown (being part of the integrated decision), hence the VS sub-problem feasible space consists of all feasible vehicle schedules that can be constructed from the whole input set of trips $\mathcal{T}$. Indeed, a vehicle schedule is, from a combinatorial point of view, a *sequence* of trips such that two subsequent ones are compatible according to the given VS constraints.

In the following, we will actually present *two* different VS graphs, which attain different trade-offs between $|N^{VS}|$ and $|A^{VS}|$ (basically, the number of linear constraints and variables in the corresponding MILP sub-model). Common to both versions is that N^{VS} contains *two* nodes i^- and i^+ for each trip $i \in \mathcal{T}$, representing the start and the end of trip i, respectively.

4.2.1. "Pure" compatibility graph

In the first variant, besides the previously mentioned trip beginning and ending nodes, G^{VS} only contain two further nodes O^- and O^+, whose out-going and in-going arcs respectively represent a vehicle performing the first pull-out and the last pull-in trips of the corresponding schedule. As for A^{VS}, it contains six types of arcs:

1. *Trip arcs* (i^-, i^+) for each trip $i \in \mathcal{T}$ (red arcs in Fig. 3), with capacity 1 and 0 cost; a unit of flow on the arc means that the corresponding trip i is "covered" in the vehicle schedule.
2. *In-line compatibility arcs* (i^+, j^-) for each pair of trips i and j that are in-line compatible (blue arcs in Fig. 3); a unit flow on the arc means that the bus covering trip i will be waiting at the terminal $en(i) = sn(j)$ and then start trip j. These arcs have capacity 1 and cost proportional to the *extra waiting time* $st(j) - et(i) - \underline{\delta}_{en(i)}^{h(i)}$ w.r.t. the minimum waiting time at $en(i)$ in the given time window.
3. *Out-line compatibility arcs* (i^+, j^-) for each pair of trips i, j that are out-line compatible (green arcs in Fig. 3); a unit of flow on the arc means that the vehicle covering trip i will perform a pull-in trip from $en(i)$ in time window $h(i)$, then perform a pull-out trip to the $sn(j)$ in time window $h(j) \geq h(i)$, then finally start performing trip j. These arcs have capacity 1 and cost proportional to $t_{en(i)+}^{h(i)} + t_{sn(j)-}^{h(j)}$, i.e., the sum of the *pull-in/out travel times* in the corresponding time windows. Note that waiting time at the depot is not penalized, because it is not covered by staff.
4. *Start arcs* (O^-, i^-) for each trip $i \in \mathcal{T}$ (dotted arcs in Fig. 3); a unit of flow on the arc means that a vehicle will perform a pull-out trip to $sn(i)$ as the first activity of its vehicle block. These arcs have capacity 1 and cost proportional to the pull-out time $t_{sn(i)-}^{h(i)}$.
5. *End arcs* (i^+, O^+) for each trip $i \in \mathcal{T}$ (also dotted arcs in Fig. 3); a unit of flow on the arc means that a vehicle will perform a pull-in trip to return to the depot from $en(i)$ as the last activity of its vehicle block. These arcs have capacity 1 and cost proportional to the pull-in time $t_{en(i)+}^{h(i)}$.
6. *Return arc*, the single (O^+, O^-) (omitted in Fig. 3). This is added in order to allow any number of units of flow, i.e., vehicles, to depart from O^- and reach O^+, thereby being used to define the vehicle schedule. By setting all *deficits* of

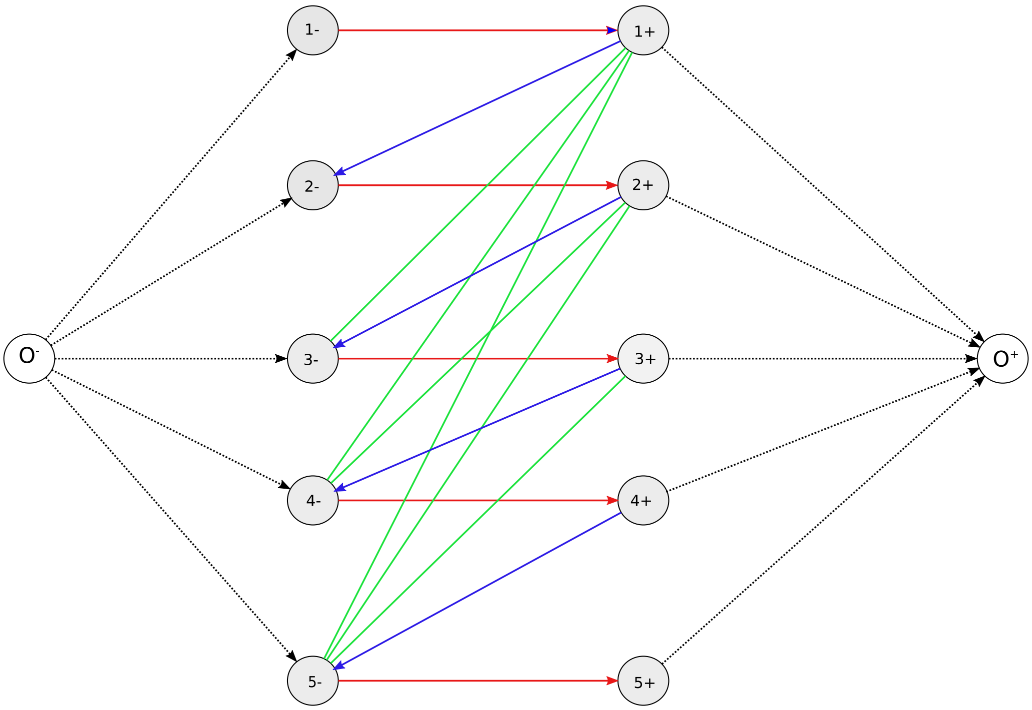

Fig. 3. G^{VS} “pure” compatibility graph for the example.

Table 1
A VS example.

i	Direction	$st(i)$	$et(i)$	$sd(i)$	$ed(i)$
1	$\vec{AB}$	7:00	8:30	6:45	9:15
2	$\vec{BA}$	9:00	10:30	8:45	11:15
3	$\vec{AB}$	11:00	12:30	10:45	13:15
4	$\vec{BA}$	13:00	14:30	12:45	15:15
5	$\vec{AB}$	15:00	16:30	14:45	17:15

the nodes to 0, this defines a *circulation problem* on the VS graph. The arc has capacity equal to the maximum fleet cardinality (if any, $+\infty$ otherwise) and a “large” cost (w.r.t. those that can typically be expected on the other types) representing the cost of using one more vehicle in the vehicle schedule.

Example 2. Consider the set $\mathcal{T}$ formed of the 5 trips described in Table 1. For simplicity, each trip lasts 90 min, travel times from/to the depot to/from both terminals are all equal to 15 min, and all minimum stopping times are 30 min. In the table, for each trip i we report the corresponding direction (i.e., either $\vec{AB}$ or $\vec{BA}$), its start and end time $st(i)$ and $et(i)$, and the *start/end depot time* instants $sd(i)$ and $ed(i)$. These are respectively the last instant in which a vehicle can start a pull-out trip in time to reach $sn(i)$ and perform the trip i ($sd(i) = st(i) - t^{h(i)}_{sn(i)-}$), and the first instant in which a vehicle, after having performed a pull-in trip from $en(i)$ at the end of trip i, is ready to leave again the depot ($ed(i) = et(i) + t^{h(i)}_{en(i)+} + \underline{\delta}^{h(i)}_{O}$). Fig. 3 shows the “pure” compatibility graph for the example.

The issue with this variant of G^{VS} is that it has a rather large number of out-compatibility arcs. Indeed, if a trip ends rather early (with respect to the planning horizon), it is likely to be out-compatible with most of the subsequent trips. We can reduce the number of arcs constructing an alternative VS graph as follows.

4.2.2. Compatibility/time-space graph

To remove all the out-compatibility arcs, we can introduce “time-depot” nodes O_t for properly chosen time instants t. In particular, for each trip $i \in \mathcal{T}$ we will define the *start-time-depot* $O_{sd(i)}$ and *end-time-depot* $O_{ed(i)}$, with the start/end depot time instants $sd(i)$ and $ed(i)$ having been defined in Example 2. We denote by $\bar{T}$ the set of time instants corresponding to all the start/end time-depot nodes in G^{VS}. Next, after having removed the out-line compatibility arcs we add the following arcs:

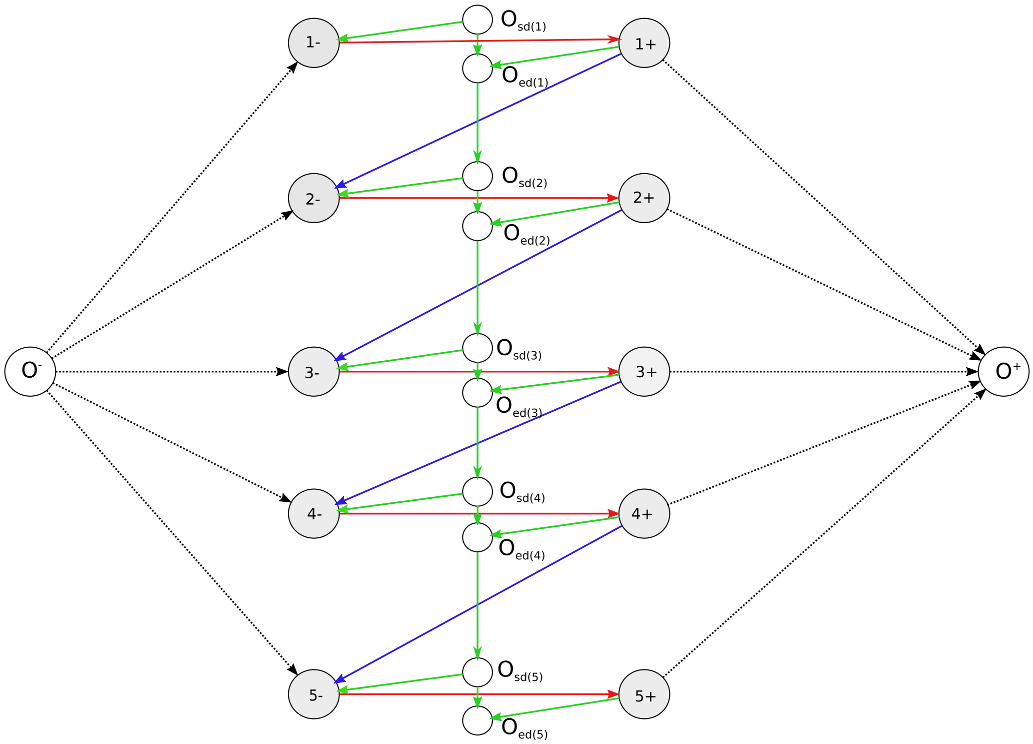

Fig. 4. G^{VS} compatibility/time-space graph.

- *Time arcs* (O_t, O_{t+1}) for all pairs $(t, t+1)$ of time instants in $\bar{T}$, where $t+1 = \min\{t' \in \bar{T} \,:\, t' > t\}$ (vertical green arcs in Fig. 4). These are the typical "holding arcs" in time-space graphs, representing the fact that all vehicles at the depot at t that have not just started a pull-out trip will remain at the depot until $t+1$. The cost of these arcs is 0 and the capacity is set equal to the maximum fleet cardinality (if any, $+\infty$ otherwise).
- *Pull-in arcs* $(i^+, O_{ed(i)})$ for all $i \in \mathcal{T}$ (diagonal green arcs in Fig. 4), representing the fact that the vehicle having just performed trip i performs a pull-in to the depot, where it arrives at time $ed(i)$. These arcs have capacity 1 and cost proportional to the pull-in time $t^{h(i)}_{en(i)+}$.
- *Pull-out arcs* $(O_{sd(i)}, i^-)$ for all $i \in \mathcal{T}$ (diagonal green arcs in Fig. 4), representing the fact that the vehicle performs a pull-out trip at time $sd(i)$, i.e., just in time to subsequently start trip i. These arcs have capacity 1 and cost proportional to the pull-out time $t^{h(i)}_{sn(i)-}$.

Basically, in this version the depot nodes O^- and O^+ are expanded in a space-time (line) graph representing the status of the depot (number of vehicles available there) at all possible times where it may change; this is why we dub it a "compatibility/time-space graph". In this version, each out-compatibility arc between two trips i and j is replaced by the i^+–j^- path consisting of a pull-in arc from i^+ to $O_{ed(i)}$, a sequence of time-arcs connecting the time-depot nodes $O_{ed(i)}$ and $O_{sd(j)}$, and a pull-out arc from $O_{sd(j)}$ to j^-. The advantage of this version is that of replacing the potentially $O(|\mathcal{T}|^2)$ out-line compatibility arcs with $O(|\mathcal{T}|)$ new nodes and arcs. Note that for some $i \neq j$, one may have $ed(i) = sd(j)$, which means that there may be strictly less than $2|\mathcal{T}|$ nodes O_t (and that, unlike in Fig. 4, these nodes can have more than three incident arcs). In our experiments, the compatibility/time-space graph has usually outperformed the "pure" compatibility one. As a final remark, if deadheading *not touching the depot* is allowed, then the corresponding *deadhead arcs* must be added to A^{VS} (in either version) that are analogous to out-line compatibility arcs save for not contemplating a return to the depot. This may seem to run contrary to the rationale of the compatibility/time-space graph, but in practice this is likely to be only possible for relatively few pairs of trips for which $en(i)/sn(j)$ and/or $et(i)/st(j)$ are "rather near", in that otherwise it is more reasonable (or required by regulations) to return to the depot anyway. Thus, the compatibility/time-space graph may also be a sensible choice in such case.

Whatever the chosen version, if the VS subproblem could be solved independently—but it can not—then it would be a min-cost circulation problem, i.e., a *minimum cost network flow (MCF)* on G^{VS}, which is polynomially solvable. Actually, the optimal solution to the VS sub-problem would trivially be the *zero circulation*, as the arc costs are non-negative and the node deficits are zero. In fact, it is due to the linking constraints described in the next sub-section, that flow will be forced to traverse the trip-arcs corresponding to the trips $\mathcal{T}^*$ selected by the TT sub-problems, and therefore produce a non-zero circulation (vehicle schedule Ω^*).

It should be remarked that the VS subproblem discussed in this section may not be capable of expressing some constraints on the vehicle routes that may be necessary in certain cases, such as those depending on the total time/distance travelled by the vehicle (refuelling, cleaning, servicing,...). Yet, some constraints on the vehicle schedules can indeed be represented by appropriate modifications to either the TT graphs or the VS graph; examples are provided in the next Section 5. Furthermore, it is in principle possible to replace the graph-based VS model with any more expressive one, e.g. based on set partitioning formulations, without changing the overall structure of the integrated model (except, very possibly, making the ITTVS even more difficult to solve in practice).

4.3. TT-VS integrated model

The integrated model combines the VS graph (in whatever version) and the TT graphs for all directions $d \in \mathcal{D}$ to yield the following MILP model:

$$\min \ \alpha c x + \sum_{d \in \mathcal{D}} c^d y^d \tag{1}$$

$$\sum_{(m,n) \in A_d^{TT}} y_{m,n}^d - \sum_{(n,m) \in A_d^{TT}} y_{n,m}^d = b_n^d \qquad n \in N_d^{TT},\ d \in \mathcal{D} \tag{2}$$

$$y_{n,m}^d \in \{0, 1\} \qquad (n, m) \in A_d^{TT},\ d \in \mathcal{D} \tag{3}$$

$$\sum_{(m,n) \in A^{VS}} x_{m,n} - \sum_{(n,m) \in A^{VS}} x_{n,m} = 0 \qquad n \in N^{VS} \tag{4}$$

$$0 \le x_{n,m} \le u_{n,m} \qquad (n, m) \in A^{VS} \tag{5}$$

$$\sum_{(n,m) \in B(i)} y_{n,m}^{d(i)} = x_{i^-,i^+} \qquad i \in \mathcal{T} \tag{6}$$

The MILP formulation is clearly formed of three distinct blocks. Constraints (2) are the flow conservation constraints of the TT subproblems, where $y_{n,m}^d$ are the arc flow variables on A_d^{TT}; the deficits b_n^d for TT are all 0 except for $n \in \{O_d^- \ O_d^+\}$, which, together with (3), ensures that the solution describes a path between O_d^- and O_d^+ in G_d^{TT} (timetable π_d). Similarly, $x_{n,m}$ are the arc flow variables on A^{VS}, and (4) the corresponding flow conservation constraints describing a circulation (vehicle schedule Ω) in G^{VS}. The capacities $u_{n,m}$ are all 1 except that of the return arc (O^+, O^-) and the time arcs (if any); the variables need not be declared integer, once this is done for the y^d, due to the total unimodularity of flow constraints. Finally, (6) are the *linking constraints* ensuring that a trip is performed in the VS if and only if it is chosen by the corresponding TT. In G_d^{TT}, trips correspond to nodes, whereas in G^{VS} trips correspond to (trip) arcs. In TT, a trip i is selected if and only if the corresponding node i belongs to the path in G_d^{TT}, i.e., there is one arc entering node i (and of course one arc leaving by flow conservation). In particular, $B(i)$ is the backward star (i.e., in-going arcs) of the node in $N_{d(i)}^{TT}$ corresponding to trip i, i.e., the set of all arcs in $A_{d(i)}^{TT}$ entering it. In VS, trip i is selected if and only if the corresponding trip arc (i^-, i^+) belongs to the circulation in G^{VS}. So the linking is obtained by imposing that the number of arcs entering node i in TT equals the flow on the trip arc (i^-, i^+) in VS. As expected, this formulation of ITTVS is $\mathcal{NP}$-hard. Indeed, even if the maximum cardinality of the fleet was one, the problem would correspond to a *Quadratic Shortest Path* problem, which is APX-hard even if only adjacent arcs play a part in the quadratic objective function Rostami et al. (2018).

The bi-objective nature of the integrated TT-VS problem is modeled using the *weighted objective function* (1), where the VS objective is scaled by a coefficient α representing the decision-maker preferences in terms of priority between the two objectives. As already remarked, the VS costs c already are obtained by properly weighting one main VS objective with two minor ones, which means that constructing (1) requires properly choosing no less than three scaling parameters. Of course, the main one is α, governing the compromise between the two contrasting objective functions of the problem (service quality vs. service provider cost). The experience of *MAIOR* personnel has been instrumental in properly setting these weights.

5. Extensions

We now describe two extensions of the models presented in the previous Section that allow to deal with nontrivial constraints on either the TT or the VS by properly modifying the underlying graphs. This is done primarily to show the flexibility of our approach and its capacity to be adapted to the different needs of different service providers, which is one of the defining technical capabilities that makes *MAIOR* a global player in its market.

5.1. Complex lines

The first extension that we consider is that of a *complex single line*, which has two *sets* of terminals $\mathcal{A}$ and $\mathcal{B}$; each trip has the form either $A_i \vec{B}_j$ or $B_j \vec{A}_i$ for $A_i \in \mathcal{A}$ and $B_j \in \mathcal{B}$. Thus, each direction of the line is actually composed of more than

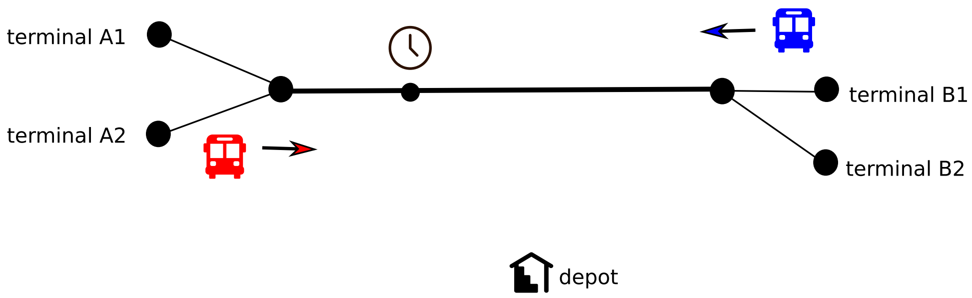

Fig. 5. "Double Y" complex single line.

one *pattern*, corresponding to different choices of the terminals. Crucially, all the patterns have to share a common central segment, where the main-stops are located, as depicted in Fig. 5 for the so-called "double Y" topology. Indeed, the headway for a direction is computed as the time separating two consecutive trips i and j at the main-stop running in that direction, *irrespectively from the pattern* they belong to, i.e., from the specific starting terminal in $\mathcal{A}$ and ending one in $\mathcal{B}$. Accounting for this case is actually simple enough provided that trips from different pairs of terminals *follow a regular scheme*. Indeed, for a complex line, service providers typically specify how the trips of the line alternate between different pairs of terminals, for instance by a simple *departure sequence scheme* σ^d and *arrival sequence scheme* σ^a, as illustrated in the following example.

Example 3. Consider the double Y topology, shown in Fig. 5, with departure scheme $\sigma^d = (A_1\ A_2)$ and arrival scheme $\sigma^a = (B_1\ B_1\ B_2)$. This means that trips must alternate first one vehicle departing from A_1 and then one departing from A_2, and a vehicle arriving in B_2 after two consecutive ones arriving in B_1. Since the departure scheme has length 2, while the arrival scheme has length 3, the complete departure-arrival sequence scheme $\sigma = (A_1 - B_1\ A_2 - B_1\ A_1 - B_2\ A_2 - B_1\ A_1 - B_1\ A_2 - B_2)$ has length 6, and repeats indefinitely along all the planning horizon.

Therefore, it is only necessary to modify the TT subproblem to account for the fact that the trips have to follow the scheme σ. In fact, nodes of G_d^{TT} are trips, i.e., pairs of terminals; arcs can therefore be seen as having the general form $(sn(i) - en(i)\ sn(j) - en(j))$ (although, of course, the time of the trip also plays a role). Ensuring that the chosen path follows the right sequence basically only amounts at removing compatibility arcs between nodes that violate it, although they would be feasible in terms of the corresponding headway. However, this would work on the original G_d^{TT} only if each *trip type* (oriented pair of terminals) appeared at most once in σ; yet, as our example shows, this is not necessarily true. Hence, one also needs to keep track of the *position in* σ of the current node $(sn(i) - en(i))$, in order to construct the correct compatibility arcs. This can be done by replicating it for each of its occurrences in σ, as the example below further illustrates.

Example 3 (continued)**.** In our example, $(A_1 - B_1)$ appears twice in σ, so we need to replicate all the nodes of this type twice to recognise whether it is the first or the second occurrence of $(A_1 - B_1)$ in σ. The same holds for $(A_2 - B_1)$. A slice of the corresponding modified compatibility graph G_d^{TT} is shown in Fig. 6, where the duplicated nodes (trips) are highlighted in red; the superscript indicates the position in the sequence. Note that, for simplicity, we only drew the arcs from one time instant to the next one, but the same arcs should be replicated for all pairs of (frequency-wise) compatible time instants.

Of course, this comes at a cost of a possibly considerable increase in the number of nodes of G_d^{TT}, although the arcs do not grow quite as rapidly because the construction is precisely aimed at removing those arcs that do not follow the right order. However, the length of σ, and therefore the size of G_d^{TT}, may grow rather rapidly as $|\mathcal{A}|$ and $|\mathcal{B}|$, and/or the complexity of σ^d and σ^a, increase. Yet, there are not too many different complex line topologies that appear in practice (the "double Y" being almost, but not quite, the most complex that can reasonably happen), nor there is usually reason to have particularly complex schemes σ^d and σ^a.

In all this, the VS model is completely unaffected. We will next present an "orthogonal" modification that rather involves G^{VS} only, leaving the G_d^{TT} unchanged.

5.2. Vehicle flow control

Public transport planners are often able to accurately estimate the number of vehicles required for different *periods* of the planning horizon, which may or may not coincide with the time-windows defined in the previous sections. The number of vehicles required depends on the expected number of passengers, the capacity of the vehicles and the frequency of service in the given periods. As already mentioned earlier, vehicle capacity is not modeled explicitly in the VS part, yet controlling the number of vehicles per time-window is another way to take it into account. Indeed, these estimates can be so accurate that the planner may want to *fix the number of vehicles per period* on input. This is called *Vehicle Flow Control* (VFC) in MAIOR, and it is relatively easy to do by modifying the graph G^{VS}, in particular in its compatibility/time-space variant (cf. Section 2.2); indeed, as we already observed, the capacity of specific arcs in G^{VS} can be used to set a limit on the number of vehicles, so

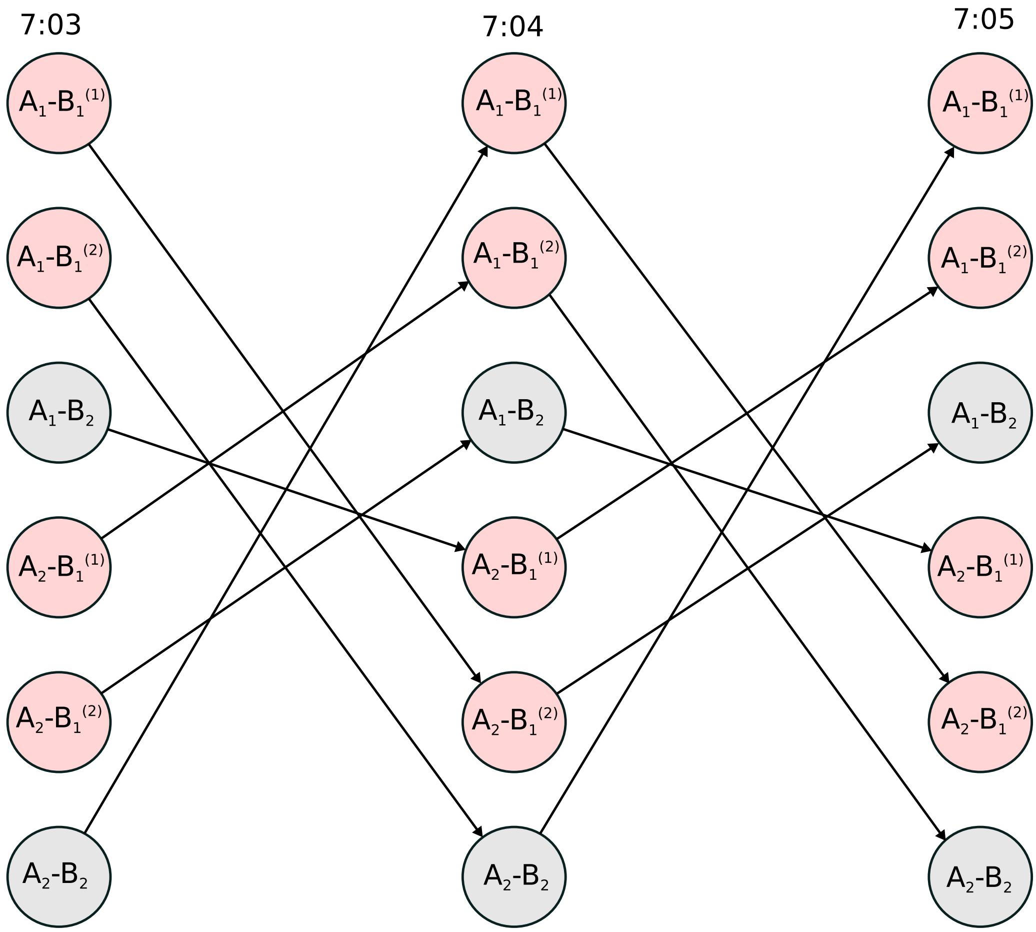

Fig. 6. A slice of the modified G_d^{TT} graph for the scheme σ.

it is not hard to bring this idea further and actually *fix* the actual number of vehicles by, basically, *fixing* the flow on some arcs. More specifically, for each of the periods $h = 1, \ldots, r$, we define $\phi(h)$ to be the number of vehicles to be fixed, and we add to N^{VS} a *local source node* O_h^- and a *local sink* node O_h^+. These nodes are given deficits that depend on the number of vehicles estimated for the corresponding period and the following/preceding period (with the right sign), as detailed below:

- For O_1^-, $-\phi(1)$, i.e., (minus) the desired number of vehicles for the first period;
- For O_h^-, $-\max\{0\ \phi(h) - \phi(h-1)\}$, for $h = 2, \ldots, r$, i.e., (minus) the difference between the number of vehicles circulating in period h and those circulating in period $h-1$ if this is positive, i.e., new vehicles have to enter in period h;
- For O_h^+, $\max\{0\,,\ \phi(h) - \phi(h+1)\}$, for $h = 1, \ldots, r-1$, i.e., the difference between the number of vehicles circulating in period $h+1$ and those circulating in period h if this is negative, i.e., vehicles have to leave after the end of period h;
- For O_r^+, $\phi(r)$, i.e., the desired number of vehicles for the last period.

This means that the VS sub-problem is no longer a circulation one, since it has as many source/sink pairs as there are periods (the deficit of all other nodes remains 0), and in fact the return time arc is also removed. Finally, arcs (O_i^-, i^-) and (i^+, O_h^+) (with 0 cost and unitary capacity) are added for all trips i belonging to period h. This construction is illustrated in Fig. 7 for the same fragment of the (compatibility/time-space) graph G^{VS} of Fig. 4, assuming there are two periods 7:00–12:30 and 12:30–16:00 in input with 1 and 2 vehicles *fixed*, respectively.

A benefit of this construction is that the primary VS cost component (i.e., the number of vehicles) is now fixed, since the number of vehicles is so. Indeed, in G^{VS} the cost was associated to the return time arc, which has now disappeared. In practice, it has been observed that this may make it somewhat easier to find good values for the weighting parameter α in the objective function.

6. Solution approach

Real-life MILP instances of ITTVS as defined in the previous section are too hard to be solved directly using a general-purpose solver within the time constraints dictated by the planners' operational requirements (this roughly means 15 min

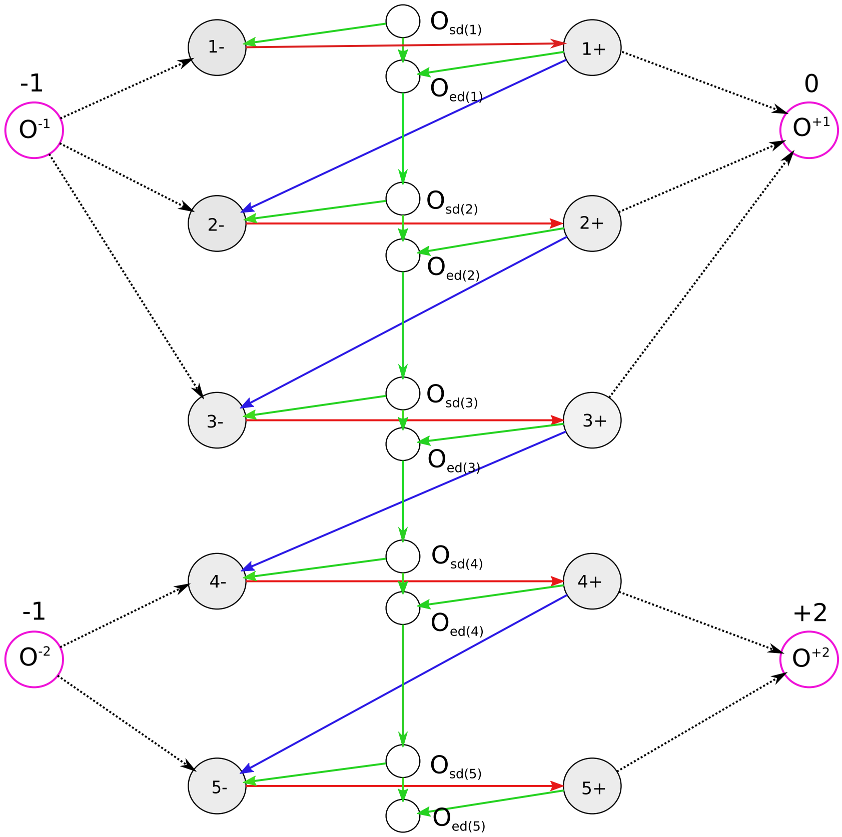

Fig. 7. Modified G^{VS} graph with VFC.

for simple lines and "a few hours" for complex lines, depending on the complexity of the scheme). Therefore, a heuristic approach is needed. We now describe a *matheuristic*, called TTD (*TimeTabling Design*), based on the solution of the continuous relaxation of (1)–(6) and on a classic *diving approach*, that at each iteration fixes "some" trips and progressively constructs a feasible ITTVS solution. The fixing is basically greedy, in that decisions taken at one iteration are usually not changed in subsequent ones, although a very limited amount of backtracking is allowed when infeasibility of the choices is detected. The iterative process runs until a complete integer solution is obtained, which basically means that for all $d \in \mathcal{D}$ the corresponding π_d forms a complete $O^-_d - O^+_d$ path, or infeasibility is detected that the backtracking is not able to resolve.

The two relevant technical aspects of the approach are how fixing is performed, and how the solution of the continuous relaxation is computed; these are basically orthogonal, and therefore are separately discussed.

6.1. Fixing strategy

The fixing strategy is based on the value of a continuous solution $(\tilde{x}\ \tilde{y} = [\tilde{y}^d]_{d\in\mathcal{D}})$, irrespectively on how this has been computed. Fixing is clearly the crucial aspect of a diving heuristic, and extensive experiments were necessary to find a fixing strategy that is both robust and efficient in practice. The best performing fixing strategy among the ones we tested turned out to be the so called *V-fix* one. On the outset, the idea is simple: to fix the trips, we first sort them in descending order of the continuous solution value $(\tilde{x}\ \tilde{y})$. However, note that, in our model, each trip $i \in \mathcal{T}$ is associated to *two* continuous solution values: $\tilde{y}_i$ in the corresponding TT, and $\tilde{x}_{(i^-,i^+)}$ in the VS. Which of the two is chosen depends on the particular *stage* the fixing rule is in, as described below. Indeed, a crucial component for the effectiveness of the fixing rule is to carefully restrict the set of trips that we consider as candidates for being fixed depending on the previous history, as we now detail. For simplicity, the description is limited to the simple single line case, and we also assume to know that the very first trip has to start from terminal A going towards B, rather than vice-versa, which is usually quite clear to planners. However, the fixing rule can be extended to complex single lines and beyond.

At the first iteration, when no trips are fixed, we select the direction $d = \vec{AB}$, and we restrict the set of candidates to the trips in the forward star (i.e., out-going arcs) of the corresponding dummy source node O^-_d in the corresponding TT subgraph G^{TT}_d, ordering them in terms of the values of the corresponding $\tilde{y}^d_i$. At the second iteration, with just one trip $\bar{i}$ fixed, we rather select the opposite direction $d = \vec{BA}$ and we restrict the set of candidates to the trips in the forward star of node $\bar{i}^+$

in the VS graph, i.e., the node representing the possible ways to chose an activity for the vehicle having just performed trip i (an in-line compatible trip, an out-line compatible trip, or a return to the depot), ordered by the corresponding $\tilde{x}_{(i^-,i^+)}$ variable instead. These two initial fixings form a "V" in the time-space graph used by transport planners to represent a timetable, whence the name.

In the subsequent iterations, we restrict the set of candidates to the union of the trips in both the forward star and the backward star in the TT graphs of the trips that have been fixed, sorted in descending order of the corresponding $\tilde{y}_i$. We then proceed at fixing the one with highest value, provided that a reasonable balance is kept between the number of trips fixed for each direction. That is, if the difference is less than 20% we allow selecting the trip to be fixed irrespectively of the direction d to which it belongs, otherwise we only select trips for the direction with fewer fixings. We sketch a very simple example to illustrate the idea of the *V-fix* strategy.

Example 4. Assume we have a simple line with terminals A and B. We start our fixing with direction $d = \vec{AB}$ and the corresponding TT graph $G^{TT}_{\vec{AB}}$. Then, we consider the out-going arcs (O^-_d, i) of the source node O^-_d, denoted by $(O^-_d, 1)$, $(O^-_d, 2)$, $(O^-_d, 3)$, $(O^-_d, 4)$, $(O^-_d, 5)$. Each arc has an associated $\tilde{y}$ (fractional) value: $\tilde{y}_1 = 0.1$, $\tilde{y}_2 = 0.2$, $\tilde{y}_3 = 0.3$, $\tilde{y}_4 = 0.35$, $\tilde{y}_5 = 0.05$, respectively. We fix the trip with largest fractional value: trip 4 in our example. Next, we move to the VS graph G^{VS} and we consider the out-going arcs $(4^+, j^-)$ of the end-trip node 4^+, denoted by $(4^+, 7^-)$, $(4^+, 8^-)$, $(4^+, 9^-)$ in our example (these can be either in-line or out-line compatible arcs). Again, each arc has an associated $\tilde{x}$ (fractional) value: $\tilde{x}_{(4^+,8^-)} = 0.3$, $\tilde{x}_{(4^+,9^-)} = 0.5$, $\tilde{x}_{(4^+,7^-)} = 0.2$. We fix the trip with largest fractional value: trip 9 in our example. These two fixings form the so-called *V-fix*.

Note that, in order to further reduce the problem size, when we fix a trip i to one (as belonging to the solution), we also fix to zero (as to not belonging to the solution) the trips belonging to the "neighborhood" of i that would violate the TT headway if they were selected, because they are "too close" in time to i. After all these fixings, the problem may have become infeasible; before confirming the fixing we check that this has not happened by solving the corresponding TT and VS subproblems, and in case we backtrack on the decision and move to the next candidate in the list. If the list becomes empty, without any (locally) feasible fixing having been identified, we accept failure and we exit from the heuristic (although this has never happened in practice, after that the fixing rule has been properly tuned). For the sake of completeness, we also mention two (now "deprecated") fixing strategies that we tested, but proved to be less effective than the one currently in use.

- *Basic diving heuristic*: The first fixing rule that we implemented. We call it "basic" because the fixing is performed by selecting the trip with largest (fractional) value in the current continuous solution $\tilde{y}$ and considering *all the trips* as possible candidates. In other words, this rule does not restrict the candidate trips to the out-going arcs of the currently fixed ones.
- *VS diving*: This is similar to *V-fix*. The only difference is that, as the name suggests, the fixing is guided by the VS continuous solution $\tilde{x}$ instead of the TT continuous solution $\tilde{y}$.

6.2. Continuous solution

We consider two ways to find a continuous solution for (1)–(6): A general-purpose LP solver and a Lagrangian relaxation. In the former case, we relax the integrality constraints (3) and solve the corresponding LP using `Clp` (Coin-OR linear programming) The Coin-OR Linear Programming project (0000), an open-source LP solver written in `C++`. Implementing the fixing in this case is trivial by just changing the bounds on the affected variables. Besides fixing to 1 the lower bound of variables corresponding to trips that are chosen to be a part of the current partial solution, we also fix to 0 the upper bound of variables representing trips that cannot possibly be chosen together with that, as discussed above.

The alternative is to use Lagrangian techniques. This corresponds to defining the vector of *Lagrangian multipliers* $\lambda = [\lambda_i]_{i\in\mathcal{T}}$ associated to the *linking constraints* (6), and solve the corresponding *Lagrangian relaxation*

$$P(\lambda) \quad \min \left\{ \alpha cx + \sum_{d\in\mathcal{D}} c^d y^d + \sum_{i\in\mathcal{T}} \lambda_i \left[\sum_{(n,m)\in B(i)} y^{d(i)}_{n,m} - x_{i^-,i^+} \right] : (2) - (5) \right\}.$$

Clearly, $P(\lambda)$ actually consists of $|\mathcal{D}| + 1$ independent sub-problems, a $TT_d(\lambda)$ for each $d \in \mathcal{D}$ and a single $VS(\lambda)$, that can be solved separately, respectively, as acyclic SPs and a MCF on the corresponding graphs G^{TT}_d and G^{VS}, as discussed on the previous sections. Note that the integrality constraints (3) are not an issue since all the individual sub-problems have the integrality property; thus, for any choice of λ, the corresponding Lagrangian relaxation solution is integral.

It is well-known that for each choice of λ, $P(\lambda)$ is a relaxation of *ITTVS*, i.e., $\nu(P(\lambda)) \le \nu(ITTVS)$ ($\nu(\cdot)$ denoting the optimal value of an optimization problem). To find the best possible Lagrangian relaxation, one then has to solve the *Lagrangian Dual*, i.e., maximize the *Lagrangian function* $\nu(P(\lambda))$ over all $\lambda \in \mathbb{R}^{|\mathcal{T}|}$. Even with the best possible choice λ^* of the Lagrangian multipliers, there is no guarantee that the penalty term in the objective function will lead to a feasible integer solution in $P(\lambda^*)$, i.e., one that satisfies the linking constraints (6). However, it is well-known that the Lagrangian Dual is equivalent to the *convexified relaxation* of the original problem; since in our case all the subproblems have the integrality property, this is actually the same as the continuous relaxation Frangioni (2005) of ITTVS, as solved by the previous approach. Thus, the

Table 2
Instance data.

Instance	T (hh:mm)	$\lvert N\rvert$	$\lvert\sigma\rvert$	I^h (mm:ss)	$\underline{\delta}_n^h$ (m)	$\overline{\delta}_n^h$ (m)	$t_{n\pm}^h$ (m)
2Cap_R1	05:30 24:00	2	1	11:00	3	10	20
2Cap_R2	05:30 24:00	2	1	12:30	3	30	21
2Cap_R3	05:00 24:00	2	1	12:00	3	10	20
2Cap_F1	05:15 22:50	2	1	13:30	2	20	18
2Cap_F2	05:50 20:40	2	1	23:00	2	10	10
2Cap_F3	05:20 24:20	2	1	18:00	2	15	9
2Cap_F4	07:00 20:25	2	1	09:00	2	10	17
3Cap_F5	05:45 24:08	3	2	08:00	2	15	18
3Cap_F6	06:34 20:53	3	2	07:30	2	10	18
3Cap_F7	05:24 22:44	3	4	08:00	2	18	17
3Cap_M1	05:57 20:31	3	2	11:00	2	13	13
4Cap_F8	06:40 20:24	4	6	06:00	2	17	16

two approaches provide the same lower bound, and thereby arguably continuous solution of “the same quality”. However, this does not mean that the time required to find the continuous solution is the same, and that the solutions themselves are necessarily identical.

Both aspects (time and solution) obviously depend on the specific algorithm used to solve the Lagrangian dual; in our case we are using an implementation of the proximal Bundle approach already used with success in other applications (e.g., Frangioni et al., 2005; Frangioni and Gorgone, 2014). Besides finding the optimal Lagrangian multipliers vector λ^*, the Bundle method also allows to explicitly construct the optimal primal solution $(\tilde{x}\,\tilde{y})$. Technically, this is done by collecting the (integer, unfeasible) primal solutions generated at each iteration, out of which the (continuous, feasible) $(\tilde{x}\,\tilde{y})$ is generated via *convex multipliers* that are automatically produced by the *master problem* solved at each iteration. Although the exact details are well-known and can be found in Frangioni (2002, 2005), it can be useful to remark that while a *feasible* $(\tilde{x}\,\tilde{y})$ is only produced at the last iteration, the algorithm produces an *unfeasible* primal solution at each iteration, and (roughly speaking) the “degree of unfeasibility” quickly decreases as the algorithm proceeds. Clearly, for the purpose of driving the fixing strategy, which is the only use of $(\tilde{x}\,\tilde{y})$ in our setting, there is no strong requirement that the solution be feasible. In this sense, our diving heuristic can be considered as well a *Lagrangian heuristic*, where the solutions of the Lagrangian subproblems are used to guide the construction of a feasible solution for the original problem; the use of the (not necessarily feasible) “convexified” primal solution in this context has already been shown to be effective in several applications Borghetti et al. (February 2003). All this allows us to explore the trade-off between terminating the algorithm early, thereby settling for a less “exact” $(\tilde{x}\,\tilde{y})$ and lower bound, but gaining in solution time, or letting it run to termination, thereby obtaining a solution $(\tilde{x}\,\tilde{y})$ with the same quality as that produced by `Clp` (albeit not necessarily exactly the same one).

Besides the approach for (iteratively) generating the Lagrangian multipliers λ, it is of course relevant how the subproblems are solved. For the TT subproblems, an hand-made implementation of the classical acyclic Shortest Path algorithm on the TT graphs G_d^{TT} is used. As for the VS subproblem, the general-purpose MCF solver `MCFSimplex` from the `MCFClass` project The mcfclass project (0000) (based, as the name suggest, on the network simplex algorithm) has been used.

A final, but important detail of the Lagrangian approach concerns how fixing is enforced in the subproblems. This is non-trivial, as several problems for which polynomial algorithms exist, among which notably shortest path ones, easily become $\mathcal{NP}$-hard if specific features are required from the solution. Fortunately, in our case this is not an issue. Indeed, for the TT subproblems we can easily fix a trip (node) as necessarily belonging to the chosen $O_d^- - O_d^+$ path by exploiting the fact that G_d^{TT} is acyclic: it is sufficient to remove all the arcs in A_d^{TT} overstepping it. Of course, fixing to zero (removing) a trip is easily obtained by removing from A_d^{TT} all arcs entering it. As for the VS subproblem, fixing to 1 a trip arc (i^-, i^+) corresponds to setting a deficit of ± 1 (with appropriately chosen signs) on its end-nodes and removing it, while fixing it to 0 just amounts at removing it (or, equivalently, setting $u_{i^-,i^+} = 0$).

7. Testing

In this section we illustrate different aspects of the performances of our approach on *real-world* instances.

7.1. Instance description

The tests have been performed on 12 real-world bus lines covering the city center of 3 major Italian cities, among which Milan. Note the tests are disjoint, i.e., each of the 12 lines is independent from the others. Indeed, as already mentioned, interlining was not considered in our case study.

This data has been provided by the corresponding bus service providers, all *MAIOR* customers. All the tests are on *single lines*, albeit possibly “complex” ones (cf. Section 5.1). Table 2 summarises the main characteristics of the instances, that are of three different types: (*i*) simple $(A - B)$ having $|N| = 2$ terminals, (*ii*) complex “Y” $(A - B_1B_2)$ with $|N| = 3$ terminals, and

Table 3
Model sizes, without vehicle flow control.

Instance	IITVS MILP			G^{VS}		G^{TT}_{AB}		G^{TT}_{BA}	
	#cols	#rows	#nnz	n	m	n	m	n	m
2Cap_R1	61,553	8886	163,030	4442	23,837	1112	18,575	1112	19141
2Cap_R2	103,820	8886	247,613	4442	66,055	1112	18,960	1112	18805
2Cap_R3	59,709	9010	157,268	4504	24,098	1113	19,094	1142	16517
2Cap_F1	90,228	8254	229,401	4126	43,333	1008	22,566	1058	24329
2Cap_F2	60,300	6894	162,279	3446	20,331	863	20,064	863	19905
2Cap_F3	100,995	8934	268,311	4466	36,894	1108	31,471	1128	32630
2Cap_F4	32,254	6382	79,746	3190	18,598	790	6701	808	6955
3Cap_F5	240,651	17,526	623,394	8762	102,893	2208	70,116	2176	67642
3Cap_F6	123,296	14,346	318,756	7172	54,557	1785	36,287	1804	32452
3Cap_F7	509,489	20,878	1,437,921	8350	94,559	4174	209,236	4180	205694
3Cap_M1	141,127	13,698	366,340	6870	60,359	1646	25,428	1770	55340
4Cap_F8	778,335	30,368	2,130,023	13,498	211,347	5018	285,804	5104	281184

Table 4
Model sizes, with vehicle flow control.

Instance	IITVS MILP			G^{VS}		G^{TT}_{AB}		G^{TT}_{BA}	
	#cols	#rows	#nnz	n	m	n	m	n	m
2Cap_R1	68,984	8904	188,402	4460	20,758	1112	24,113	1112	24113
2Cap_R2	109,544	8906	267,310	4462	63,530	1112	23,007	1112	23007
2Cap_R3	65,524	9026	177,707	4520	21,104	1113	24,113	1142	20307
2Cap_F1	191,054	8282	533,783	4154	41,429	1008	63,933	1058	85692
2Cap_F2	61,547	6900	169,282	3452	17,069	863	21,813	863	22665
2Cap_F3	170,705	8954	480,455	4486	33,880	1108	68,332	1128	68493
2Cap_F4	30,590	6402	76,988	3210	16,364	790	7032	808	7194
3Cap_F5	421,665	17,540	1,174,016	8776	95,313	2208	164,216	2176	162136
3Cap_F6	128,912	14,360	340,374	7186	49,787	1785	44,293	1804	34832
3Cap_F7	975,195	20,904	2,838,761	8376	90,837	4174	456,132	4180	428226
3Cap_M1	135,527	13,706	355,576	6878	54,323	1646	25,428	1770	55776
4Cap_F8	815,978	30,378	2,252,133	13,508	202,166	5018	322,596	5104	291216

(*iii*) complex "double-Y" ($A_1A_2 - B_1B_2$) with $|N| = 4$ terminals. All simple topologies and two of the "Y" topologies have no frequency schemes, one of the "Y" topologies has $|\sigma| = 4$, and the "double-Y" one has $|\sigma| = 6$. Time is discretized in minutes, and for each minute of the time horizon there is a possible trip i in $\mathcal{T}$ for each pattern. A typical time-horizon ranges roughly from 5:00 to 24:00, for a total of 1140 possible trips for each pattern. There are usually around 9 time windows, or about one time window each 2 h. In Table 2 we report, for each line, the time horizon "T", the number of terminals "$|N|$", the scheme length "$|\sigma|$", and the *average over the different time-windows* of ideal headways "I^h", min/max dwell times "$\underline{\delta}^h_n/\overline{\delta}^h_n$", and pull-in/out times "$t^h_{n\pm}$". Recall that there is no maximum dwell time for the depot. The length of the lines, not reported in the table as it is not part of the algorithm input, ranges from 4 to 16 Km, with an average of 11 Km.

Further insight on the data of the problem is provided by Tables 3 and 4, which report the size of the MILP model (1)–(6) for all instances. In particular, we report the number of rows, columns and non-zeros ("#rows", "#cols", "#nnz", respectively) for the whole ITTVS models, as well as the number of nodes ("n") and arcs ("m") of the graph models G^{TT}_d for the two timetabling sub-problems, and G^{VS} for the vehicle scheduling sub-problem. The two different Tables report data respectively for the "base" model of Section 3 (with the extension to "complex" lines described in Section 5.1), and for that with VFC of Section 5.2. An even more detailed breakdown of the instance data is deferred until Appendix 1.

7.2. Computational environment

We will present results obtained with two versions of our heuristic approach, as described in Section 6: "h-B" (approximately) solves the Lagrangian dual using the Bundle method, while "h-C" solves the continuous relaxation of (1)–(6) using the open-source LP solver `Clp`. We will also compare these results with those obtained by directly applying the state-of-the-art, commercial MILP solver `Cplex 12.7` on model (1)–(6). All the experiments have been performed on a PC with a 1.9 Ghz Intel Xeon (R) E5-2420 processor. For the LP relaxations of "h-C" we used the open-source `Clp` solver included in `Cbc-2.9.8`. The Bundle method was implemented in the `Bundle` solver, a C++ code developed by the second author over the years that has been repeatedly shown to be competitive for the solution of Lagrangian duals of integer programs

Table 5
h-B vs. h-C, without vehicle flow control.

Instance	h-C	h-B	h-B *vs.* h-C		
	Time	Time	Time $\phi_{\%}$	LB $\phi_{\%}$	UB $\phi_{\%}$
2Cap_R1	6	28	394	−4.28	−3.53
2Cap_R2	13	36	168	−2.64	−5.81
2Cap_R3	8	28	249	−6.76	−0.93
2Cap_F1	10	23	124	−5.07	−2.31
2Cap_F2	3	6	86	−5.57	−0.41
2Cap_F3	13	19	49	−9.75	−30.17
2Cap_F4	2	16	842	−2.72	−4.64
3Cap_F5	116	172	49	−18.32	−4.39
3Cap_F6	47	101	115	−2.56	−3.58
3Cap_F7	628	195	−69	−6.34	10.60
3Cap_M1	66	53	−20	−4.01	0.46
4Cap_F8	2410	438	−82	−7.96	4.37

(Frangioni et al., 2005; 2008), in particular with multicommodity structure as in this case (Frangioni and Gallo, 1999; Crainic et al., 2001; Cappanera and Frangioni, 2003; Frangioni and Gorgone, 2014).

All the solvers were finely tuned by performing extensive experiments in order to find a good trade-off between solution quality and running time. In particular, the parameters of `Cplex` were all set to their default values save for `CPXPARAM_Emphasis_MIP` $=$ 1 and `CPXPARAM_MIP_Strategy_LBHeur` $=$ 1. For the `Clp` solver, we used the *barrier* method at the first iteration, and the *dual simplex* in the subsequent ones. As for the `Bundle` solver, the most important parameter is the maximum number of iterations, which should be large enough to produce good quality solutions, but not too large in order to avoid to excessively increase the running time; the value providing the best compromise has been experimentally determined.

7.3. Results without vehicle flow control

We first report results for the unabridged version of our matheuristic, i.e., without VFC (but, necessarily, with the handling of "complex" lines). In Table 5 we compare the performance of the two versions, 'h-B" and "h-C' in terms of: (*i*) *Lower bounds* obtained at the very first iteration, before any fixing is done (`Clp` computes the exact optimal value of the continuous relaxation of (1)–(6), whereas the Bundle only computes a lower approximation due to being stopped early); (*ii*) *upper bounds* (value of the feasible ITTVS solution produced), and (*iii*) solution *time* (in minutes). To improve readability, we often report the *percentage difference* between two values *X vs Y*, denoted by $\phi_{\%}(X, Y)$, computed as $\phi_{\%}(X, Y) = (X - Y)/Y \times 100$.

Table 5 shows that h-C is generally faster than h-B, apart from the last three (larger, and more difficult) instances. The lower bounds computed by the Lagrangian approach are uniformly (and sometimes consistently) weaker, for two reasons: the maximum limit on the number of iterations of the Bundle algorithm, and the fact that `Clp` is used from within `Cbc`, which is a MILP solver, which generates cutting planes to strengthen the continuous relaxation of the MILP (1)–(6). However, the quality of the feasible ITTVS solutions found by h-B is usually better, apart from the last three instances. The result may seem counter-intuitive, but it has been shown in other applications Borghetti et al. (February 2003); Frangioni et al. (2005, 2008) that the $(\tilde{x}\ \tilde{y})$ solution computed by the Lagrangian approach, although not feasible, is often effective for constructing feasible solutions, possibly even more so than the standard continuous relaxation solution. The fact that this does not happen with the largest instances may be due to the iteration limit for the Bundle algorithm being uniform: possibly, on larger instances more iterations are needed to compute primal solutions of appropriate quality. Indeed, h-B being faster than h-C precisely in these instances illustrates the nontrivial trade-off between solution time and solution quality.

Next, in Table 6, we asses the performance of our heuristics in terms of solution "quality", i.e., comparing the value of the feasible solution it finds against: (*i*) The solution constructed manually, out of experience, by the expert bus planners at the service providers ("Exp"), and (*ii*) the best solution found by `Cplex` directly ran on the full MILP formulation (1)–(6) in "comparable" time. More precisely, we define the "BTS" (best time solver) for each instance as the *fastest* (hence, not necessarily more accurate) between h-B and h-C. Then, with *BT* (best time) the corresponding total running time (ranging from a few minutes to 6 h), we set the time limit for `Cplex` to *BT* ("C*1"), 2*BT* ("C*2") and 4*BT* ("C*4"). An entry "–" means that `Cplex` was not able to find any feasible solution within the time limit. The last three columns ("g*-") show `Cplex` *integrality gaps* ($\phi_{\%}$(*UB, LB*)) at termination for the three time limits considered.

The table shows that our heuristics find much better solutions than the ones obtained manually. For smaller instances, and allowing much longer times, CPLEX sometimes finds better solutions, but in general the heuristic approach is quite competitive, especially considering the many cases in which `Cplex` could not find any feasible solution within the time limit. In particular, for complex lines, a direct solution of the MILP formulation, even with a state-of-the-art MILP solver like `Cplex`, is never competitive. As a testament of the difficulty of solving the model, even allowing far larger running times `Cplex` never even gets close to solving the problem to anything resembling optimality, as the huge final gaps show.

Table 6
Expert and `Cplex` solutions vs. fastest heuristic.

Instance	BTS	ITTVS sol. $\phi_{\%}$: *vs.* BTS				`Cplex` gap $\phi_{\%}(UB, LB)$		
		Exp	C*1	C*2	C*4	g*1	g*2	g*4
2Cap_R1	h-C	117.93	–	–	4.60	–	–	50.79
2Cap_R2	h-C	28.84	1863	0.75	0.75	3488	84	83.11
2Cap_R3	h-C	35.35	1488	1488.69	0.71	2290	2290	51.51
2Cap_F1	h-C	202.99	3050	2.06	−1.50	5021	66	60.13
2Cap_F2	h-C	51.78	1102	885.69	25.72	1539	1243	71.32
2Cap_F3	h-C	158.86	5338	908.09	−16.23	12,508	2237	94.20
2Cap_F4	h-C	19.04	–	−7.88	−7.88	–	389	38.84
3Cap_F5	h-C	252.98	–	48.84	48.61	–	499	486.92
3Cap_F6	h-C	81.38	–	19.61	19.61	–	93	92.75
3Cap_F7	h-B	63.53	–	–	–	–	–	–
3Cap_M1	h-B	16.32	–	–	6.11	–	–	153.24
4Cap_F8	h-B	61.55	–	–	–	–	–	–

Table 7
Manual vs. fastest heuristic.

Instance	Exp				BTS			
	w	#*c*	*ρ*%	#*v*	*w*	#*c*	*ρ*%	#*v*
2Cap_R1	475	189	8.99	16	183	187	3.78	16
2Cap_R2	488	176	2.29	16	272	174	5.84	15
2Cap_R3	228	186	5.91	17	225	185	4.20	18
2Cap_F1	394	138	14.74	11	146	136	3.47	9
2Cap_F2	169	76	2.57	3	96	76	2.35	4
2Cap_F3	408	119	12.53	7	128	122	3.11	6
2Cap_F4	98	196	3.66	9	116	194	2.87	9
3Cap_F5	598	262	42.10	8	497	262	7.53	7
3Cap_F6	393	269	7.88	17	136	265	5.74	16
3Cap_F7	574	282	16.47	24	411	274	10.02	22
3Cap_M1	393	168	2.65	9	206	168	3.56	9
4Cap_F8	1085	279	6.10	21	497	281	8.73	20

Arguably, figuring out what the difference in objective function value actually means in terms of the transportation setting is somewhat hard. To help in this, in Table 7 we report the comparison between the solutions obtained by our matheuristic and the expert-provided ones in terms of three indicators:

1. $\rho\%$ is a *frequency regularity* index, representing the average deviation (in minutes) with respect to the ideal headway (TT objective), expressed as a percentage of the ideal headway;
2. $\#v$ is the *number of vehicles* used (primary VS cost);
3. w is the *extra waiting time* at the terminals (in minutes) with respect to the minimum waiting time required (secondary VS cost).

We also add the total *number of trips* ($\#c$) performed in the solution; although this is not directly a part of the objective function, it is an interesting operational information.

The results paint a quite favourable picture of our heuristics. Most often than not, the provided solution is just better when measured along *all* operationally significant measures. Sometimes the trade-off between the two objective functions shows off; for instance, in the largest `4Cap_F8` a slight worsening in the regularity index is paid off by a significant improvement of the waiting time, and one vehicle less. Only in the (small) `2Cap_F2` our heuristics perform somehow worse that the manual solution, requiring one vehicle more to yield similar results in the other measures. It should also be remarked that the expert solutions may be taking into account some aspects that our model does not consider, such as synchronization with other lines. However, what is important is that the solutions are of comparable quality (if not better) than those obtained manually. The role of our model in the actual Decision Support System that MAIOR provides to its customers is to automatically suggest solutions that can be taken as a good basis for the decision, possibly after having been manually modified. Thus, besides the actual measures as reported in the Table, what is perhaps most important to us has been having had the valuable support fromexperts in *MAIOR*, who regularly work with customers on these issues and therefore have a deep knowledge of the planners' preferences and objectives. They have confirmed that the solutions produced by TTD are indeed of comparable quality, and often significantly better, than those usually employed in operations. Further insight on the operational quality of the obtained solutions is provided in Appendix 2.

Of course, whenever a bi-objective problem is solved, a crucial issue is that of providing the decision-maker with easy and consistent ways of influencing the solution process in order to provide solutions with the desired trade-off between the

Table 8
h-B vs. h-C, with VFC.

Instance	h-C	h-B	h-B *vs.* h-C		
	Time	Time	Time $\phi_{\%}$	LB $\phi_{\%}$	UB $\phi_{\%}$
2Cap_R1	3	18	505.60	−0.50	0.01
2Cap_R2	6	38	485.59	−0.17	0.68
2Cap_R3	2	32	1605.99	−0.12	0.06
2Cap_F1	10	10	1.28	−0.52	−1.44
2Cap_F2	2	1	−38.61	0.00	0.00
2Cap_F3	8	10	29.78	−0.55	−0.25
2Cap_F4	1	21	1943.34	−0.11	0.08
3Cap_F5	239	305	27.44	−4.11	1.41
3Cap_F6	92	153	66.49	−0.34	0.14
3Cap_F7	2354	349	−85.18	−2.21	−1.18
3Cap_M1	23	57	142.39	−0.27	0.05
4Cap_F8	5592	1044	−81.33	−0.60	0.02

Table 9
Manual and Cplex solutions vs. fastest heuristic, with VFC.

Instance	BTS	ITTVS sol. $\phi_{\%}$: *vs.* BTS				Cplex gap $\phi_{\%}(UB, LB)$		
		Exp	C*1	C*2	C*4	g*1	g*2	g*4
2Cap_R1	h-C	5.84	–	0.00	0.00	–	0.00	0.00
2Cap_R2	h-C	5.63	–	–	−0.03	–	–	0.01
2Cap_R3	h-C	2.40	–	–	–	–	–	–
2Cap_F1	h-C	8.61	–	−2.81	−2.83	–	0.20	0.17
2Cap_F2	h-B	6.53	–	–	0.00	–	–	0.00
2Cap_F3	h-C	9.39	–	−0.36	−0.36	–	0.00	0.00
2Cap_F4	h-C	2.59	–	−0.14	−0.14	–	0.00	0.00
3Cap_F5	h-C	7.79	–	–	−2.13	–	–	3.97
3Cap_F6	h-C	2.53	–	–	50.34	–	–	52.85
3Cap_F7	h-B	9.03	–	–	–	–	–	–
3Cap_M1	h-C	4.92	–	–	–	–	–	–
4Cap_F8	h-B	4.38	–	–	–	–	–	–

two objectives. In our case, this is possible in several ways. Obviously, one can tweak the values of the scaling constants (primarily, but not only, α). Another one, specifically required by the planners, is the Vehicle Flow Control of Section 5.2, whose results are analysed next.

7.4. Results with vehicle flow control

We now analyse the performance of our TTD heuristic when the number of vehicles for each period is fixed on input (VFC), as described in Section 5.2. We consider the same set of instances used for the first case study (Table 2), the only difference being the additional information on the number of vehicles allowed to pull-in and pull-out in each time period. In Table 8 we compare the performance of the two heuristics in the same way as in Table 5, in Table 9 we asses the performance of our heuristic in terms of solution quality comparing with manual and Cplex solutions, as in Table 6 and, finally, in Table 10 we compare the "operational" quality of the solutions, as in Table 7.

Regarding the comparison between h-B and h-C, the same trends apparent in the previous results also show off here, i.e., the former is generally slower except on some complex lines, Lagrangian lower bounds are rather weaker, but its solutions are generally better. Yet, the difference in performance is considerably less marked, perhaps indicating that VFC constrains much more tightly the set of feasible solutions to the problem, preventing the heuristics to construct much better (or much worse) solutions to one another. Similarly, the comparison with Cplex and the manual solution paints a similar picture, except that the difference in solution quality is far less pronounced. Even when, given much longer running time, Cplex can find better solutions than the heuristics, the gain is rather small; again, Cplex often fails to find solutions at all, especially for complex lines. Thus, these results confirm that whenever the complexity of the problem increases, a heuristic approach is crucial as even state-of-the-art MILP solvers are not competitive. Still, the heuristics provide better solutions than the manual approach, to the tune of 2–10%; the gain is not very large, but consistent throughout the test set, indicating that the approach can reliably relieve planners with tedious and repetitive handwork. Furthermore, the detailed transportation data of Table 10 shows that the solutions hit a considerably different trade-off w.r.t. those of the non-VFC case: With the same number of vehicles (obviously, to make the comparison fair), the heuristic schedules significantly more trips, resulting in significantly decreased waiting time (in some cases by about one order of magnitude) at the cost of a significantly decreased regularity (in some cases by a factor of three or more). Yet, the solutions have been deemed to be fully satisfactory by the

Table 10
Manual vs. fastest heuristic, with VFC.

Instance	Exp				BTS with VFC			
	w	#c	ρ%	#v	w	#c	ρ%	#v
2Cap_R1	475	189	8.99	16	59	200	12.77	16
2Cap_R2	488	176	2.29	16	76	188	4.35	16
2Cap_R3	228	186	5.91	17	96	190	6.49	17
2Cap_F1	394	138	14.74	11	127	142	14.46	11
2Cap_F2	169	76	2.57	3	21	81	2.90	3
2Cap_F3	408	119	12.53	7	49	130	11.18	7
2Cap_F4	98	196	3.66	9	29	201	4.47	9
3Cap_F5	598	262	42.10	8	375	272	75.00	8
3Cap_F6	393	269	7.88	17	124	273	9.64	17
3Cap_F7	574	282	16.47	24	374	284	33.75	24
3Cap_M1	393	168	2.65	9	47	176	4.76	9
4Cap_F8	1085	279	6.10	21	437	292	19.54	21

Table 11
"zero TT", "zero VS", and "normal" solutions.

Instance	"zero TT"				"zero VS"				BTS			
	w	#c	ρ%	#v	w	#c	ρ%	#v	w	#c	ρ%	#v
2Cap_R1	2	153	32.41	6	639	190	2.64	48	183	187	3.78	16
2Cap_R2	2	143	29.70	7	1108	176	0.10	18	272	174	5.84	15
2Cap_R3	3	145	39.27	7	522	187	0.01	56	225	185	4.20	18
2Cap_F1	7	88	50.90	5	961	140	0.23	14	146	136	3.47	9
2Cap_F2	0	77	14.75	3	128	75	0.77	47	96	76	2.35	4
2Cap_F3	5	98	19.90	4	405	120	0.15	22	128	122	3.11	6
2Cap_F4	1	152	19.60	5	728	193	0.08	16	116	194	2.87	9
3Cap_F5	99	193	44.32	6	1211	268	0.13	22	497	262	7.53	7
3Cap_F6	26	190	29.99	11	1156	265	0.47	44	136	265	5.74	16
3Cap_F7	135	241	19.33	20	1297	277	0.12	33	411	274	10.02	22
3Cap_M1	30	170	9.38	9	640	169	0.49	47	206	168	3.56	9
4Cap_F8	246	228	35.15	15	1703	273	0.36	25	497	281	8.73	20

experts. This shows that our approach is capable of allowing the planners to influence the trade-off between the two aspects of the problem (objective function) in several ways, while still producing solutions that are operationally useful.

7.5. Further analysis

In this section we report the results of further analysis for our case studies. In particular, we report data that shows the "effect" of integration with respect to solutions where either *only the timetabling* or *only the vehicle scheduling* objective function is optimized, and we analyse the performance of our matheuristic on *reduced test cases* (derived from the original ones), that can be solved by `Cplex` at optimality.

7.5.1. "Zero-TT" vs *"Zero-VS"*

Here we report the results of a simple experiment aimed at giving a sense of the available trade-off between the two objective functions. In Table 11 we report on the results obtained by running our matheuristic (without VFC) with three different configurations of the costs: the standard one of Section 7.3, the "zero TT" one where the timetabling regularity objective term is almost zeroed out (i.e., α is taken "very large"), and the 'zero VS" one where VS costs objective term is almost zeroed out (i.e., α is taken "very small"). Clearly, it would make no sense to compare the objective function values of our solutions; instead, we compare the "operational" quality of the solutions with the same parameters as in Tables 7 and 10.

The results clearly illustrate the very substantial trade-off between the two measures. By entirely focussing on the service provider's objective (the "zero TT" case) a drastic reduction on the number of vehicles can be obtained, but at the cost of a very poor regularity, i.e., very large deviations from the ideal headway. On the other hand, insisting on an (almost) *periodic* timetable where regularity is (almost) never violated (the "zero VS") incurs in a much higher operating cost with respect to both the number of vehicles (primary VS objective) and the waiting times (secondary objective). The solutions of our matheuristic (BTS) indeed manage to strike a balance between the two contrasting objectives, leading to solutions whose indicators are kind of "midway" between the two extremes.

Table 12
Cplex *vs.* fastest heuristic on reduced instances.

Instance	$\phi_{\%}$	Cplex					BTS				
	o.f.	w	$\#c$	$\rho\%$	$\#v$	Time	w	$\#c$	$\rho\%$	$\#v$	Time
2Cap_R1*	−7.30	287	113	2.21	11	25,167	323	114	2.03	12	69
2Cap_R3*	−8.16	340	118	1.60	13	4821	357	118	2.05	13	59
2Cap_F1*	−4.05	258	99	3.87	9	6339	274	99	3.92	9	67
2Cap_F2*	−0.33	237	69	0.96	8	14,639	234	69	1.05	8	85
2Cap_F3*	−8.87	187	86	1.66	6	4401	229	86	1.35	7	100
2Cap_F4*	−9.24	220	122	2.39	8	6544	274	122	1.60	9	35

7.5.2. Reduced test cases

The real-world ITTVS instances considered in the previous sections are too large to be solved by Cplex at optimality. To get a better grasp of what kind of solution quality improvements we could still obtain by improving our heuristics, in Table 12 we consider 6 instances obtained from the original two-terminal simple line ones by reducing the time-horizon, so that they can be solved by Cplex at optimality. Each instance corresponds having deleted the second part of the afternoon, roughly from about 17:00 on, from the original one; details differ slightly for each in order to have the operation compatible with the original time windows, but in general each reduced instance corresponds to about 60% of the original one. For these we report the percentage difference $\phi_{\%}$ of the corresponding objective function values (column "o.f.") between the optimal (integer) solution computed by Cplex and the solution provided by BTS. We also put these in context by showing the parameters describing the solution from a transportation viewpoint as in Tables 7, 10 and 11. Finally we report the running time of the two approaches (in seconds). The results show that there is indeed further scope to improve the effectiveness of our heuristics: solutions can be found that can improve the objective value by up to about 10%, which may lead to visible improvements from the practical viewpoint, such as using one vehicle less. In general, however, solution quality can be deemed to be comparable. Furthermore, better-quality solutions come at the cost—at least when using a general-purpose (albeit state-of-the-art) solver—of more than two orders of magnitude more time, which clearly makes the approach unusable in practice. Meanwhile, the solutions provided by BTS in reasonable time are still reasonably good-quality ones.

8. Conclusions

We have presented a new model for integrated timetabling and vehicle scheduling which, given a set of possible trips and the desired headways, produces a timetable and a set of vehicle schedules with the best (user-defined) compromise between service quality (deviation from the ideal frequency of service) and service cost (number and "cost" of vehicles used). This is a problem that service planners in real-world public transportation companies face day-to-day, and that is usually manually solved through labor and experience.

Our model is based on combining compatibility graphs representations of both the TT subproblems (minimizing deviation from the ideal frequency of service) and the VS subproblem (finding a minimal-cost vehicle schedule), although for the latter a hybrid compatibility/time-space graph formulation is usually preferable. The model is quite flexible and can handle further requirements suggested by *MAIOR* customers, namely complex lines and vehicle flow control. The corresponding MILP formulation is a large-scale multicommodity-type integer program, which is hard to solve in short time with standard techniques. We have therefore proposed a matheuristic approach, that at each iteration solves the continuous relaxation of the problem, either via a general-purpose LP solver or via Lagrangian techniques, and uses the resulting information to drive a diving heuristic.

Testing the approach on real-world instances of the service providers for three major Italian cities has shown that the heuristic consistently and reliably provide solution of comparable—and most often better—quality than those constructed manually by experts, thereby indicating that the model can be used to aid even experienced planners in either obtaining better solutions, or obtaining them faster and with less effort, or both. Also, the heuristic is quite competitive w.r.t. the direct use of state-of-the-art, general-purpose solvers like Cplex.

The proposed approach lends itself to different uses in an actual operating environment. Most *MAIOR* customers that have tested it use it in a single-line setting like the one envisioned in Section 7. This makes sense in particular for high-intensity lines, possibly using dedicated vehicles (double-length buses, metros, trams,...), whereby interlining is necessarily either absent or very reduced. However, the approach can also be used in the standard sequential planning process only to construct the timetables, with the knowledge that a "good" underlying VS then exists for each single (direction of each) line, which can then be improved by a global, inter-line VS step. Indeed, the vehicle schedules produced by the approach can be passed in input to the inter-line VS solver (which is often based on column generation), ensuring that the quality of the VS for the lines where the approach has been used can only improve, exploiting interlining, w.r.t. what is possible considering each line separately.

Actually, it would conceivably be possible to run the model for the whole of a city planning, i.e., for all the lines at the same time: the corresponding problem would however be huge, and making this feasible from the computational viewpoint requires further research.

Also, it could be possible to incorporate in the model other features suggested by *MAIOR* customers, e.g. ones where specific constraints are imposed on the vehicle schedules to ease the construction of feasible crew duties in the last planning phase. Of course, this is always possible by integrating a full-blown constrained VS model in the ITTVS, but this comes at a potentially high computational cost. Alternatives might be possible where the required structures are instead modeled by clever manipulations of the graphs structure, akin to those already illustrated in the paper.

Acknowledgments

The third author gratefully acknowledges the partial financial support from the University of Pisa, project “Modelli ed algoritmi innovativi per problemi strutturati e sparsi di grandi dimensioni” (grant PRA_2017_05). The second author gratefully acknowledges the partial financial support from the Italian Ministry of Education, University and Research (MIUR), project PRIN 2015B5F27W “Nonlinear and Combinatorial Aspects of Complex Networks”. This work has also been partly funded by the Europeans Union’s EU Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie Action “Mixed-Integer Non Linear Optimisation: Algorithms and Applications” (grant agreement no. 764759). The authors are grateful to Alessandro Bertolini, M.Sc of the University of Pisa, for his help with coding an early version of the approach, as well to the bus service operators, among which *ATM* Milan, for providing the instances used for the case studies. Finally, we are indebted to the anonymous Referees for their numerous and detailed comments, which has helped us to significantly improve on the original version of the paper.

Appendix A. Detailed instance information

In this Appendix we provide more detailed information about the instances, trying to provide a better sense of the variability of the data. In particular, Table 13 describes the different time windows in which the planning horizon is subdivided. For each time window we report the right extreme (“$\overline{T}$”) of the corresponding interval; the first “empty” window is only meant to indicate where the first “real” time window starts. For the TT time windows we report the *minimum* (“$\underline{I}^h$”), *ideal* (“I^h”) and *maximum* (“$\overline{I}^h$”) headway. Note that different data is reported for the two directions (TT subproblems) $\vec{AB}$ and $\vec{BA}$, as both the number of time windows, their position and the headways are different for each, as required by the planners. We also report the time windows of the VS subproblem with the VFC feature, with the corresponding required number of vehicles (“#v”); again, the number and placement of these is different from the TT ones. For the sake of space, we only present details on four instances out of the twelve considered in the case studies, since the others present a similar pattern. The data shows the flexibility of the model, that allows the planners to make detailed and specific requirements to both the desired headways–for each direction–and the VS problem separately.

Appendix B. `TTD` Screenshots

To provide further insight on the value of allowing “small” deviations from a completely periodic timetable, we report in Figs. 8 and 9 two screenshots taken from the actual graphical interface of *MAIOR* software for the TTD module implementing the approach described in this paper. These represent timetabling corresponding to the “peak-hour” trips for the simple line instance `2Cap_R2`, whose details are reported in Table 13, with two terminals (“CBN” and “MGI”), and an ideal headway

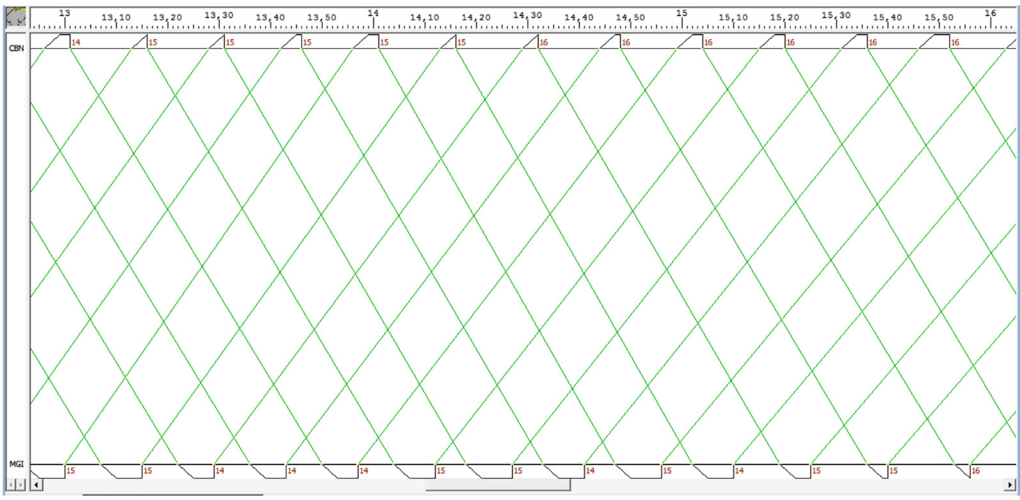

Fig. 8. Screenshot of a TTD solution at peak hours.

Table 13
More detailed instances description.

Instance	TT-$\vec{AB}$				TT-$\vec{BA}$				VS with VFC	
	$\overline{T}$	$\underline{I}^h$	I_h	$\bar{I}^h$	$\overline{T}$	$\underline{I}^h$	I_h	$\bar{I}^h$	$\overline{T}$	#v
	5:30	0	0	0	5:30	0	0	0	5:30	0
	5:51	480	960	1500	6:55	390	780	1500	6:00	4
	6:52	390	780	1500	9:57	270	540	1080	7:00	6
	8:57	300	600	1200	12:26	420	840	1500	10:35	12
	9:48	240	480	960	15:48	390	780	1500	17:15	8
2Cap_R1	13:12	420	840	1500	16:56	420	840	1500	20:35	11
	16:56	390	780	1500	19:55	270	540	1080	21:30	7
	20:01	270	540	1080	20:52	360	720	1440	22:20	5
	20:59	360	720	1440	21:44	420	840	1500	23:50	3
	24:00	420	840	1500	24:00	570	1140	1500	24:40	4
	5:30	0	0	0	5:30	0	0	0	5:30	0
	6:22	450	900	1500	6:23	480	960	1500	6:25	5
	9:59	300	600	1200	9:57	300	600	1200	6:55	7
	11:55	360	720	1440	11:53	360	720	1440	8:15	11
	16:26	450	900	1500	16:20	450	900	1500	10:50	12
2Cap_R2	18:25	330	660	1320	18:27	330	660	1320	12:40	10
	19:54	300	600	1200	19:51	300	600	1200	16:35	8
	21:04	420	840	1500	20:42	420	840	1500	20:45	12
	24:00	570	1140	1500	22:23	480	960	1500	21:35	7
	39:59	0	0	0	24:00	600	1200	1500	23:05	5
	–	–	–	–	–	–	–	–	24:40	4
	–	–	–	–	–	–	–	–	39:59	0
	5:23	0	0	0	5:27	0	0	0	5:00	0
	6:23	300	600	1200	6:10	600	1200	1800	5:40	3
	10:20	180	360	720	7:56	150	300	600	6:05	8
	13:42	210	420	840	10:38	180	360	720	6:55	19
	15:59	240	480	960	11:34	210	420	840	10:55	21
	17:00	270	540	1080	12:35	180	360	720	13:35	19
3Cap_F7	20:15	210	420	840	13:29	210	420	840	16:05	16
	20:38	270	540	1080	16:23	240	480	960	17:10	17
	22:44	540	1080	1680	20:06	210	420	840	20:20	19
	–	–	–	–	20:18	270	540	1080	21:00	14
	–	–	–	–	22:25	420	840	1800	21:30	9
	–	–	–	–	–	–	–	–	22:40	6
	–	–	–	–	39:59	0	0	0	23:30	3
	6:54	0	0	0	6:40	0	0	0	6:20	0
	8:02	150	300	660	6:48	120	240	480	7:00	15
	9:44	180	360	720	6:58	150	300	600	13:10	21
4Cap_F8	10:08	150	300	600	7:25	180	360	720	18:25	20
	12:15	180	360	720	7:38	150	300	600	20:00	19
	12:51	150	300	600	18:52	180	360	720	20:51	13
	19:22	180	360	720	19:54	210	420	720	–	–
	19:46	210	420	720	20:01	270	540	720	–	–
	20:07	240	480	720	20:20	330	660	720	–	–
	20:24	300	600	720	–	–	–	–	–	–

of 15 min in both directions, under the form of the typical time-space charts familiar to planners, where each trip is a green line. The main stop corresponds to the departure terminal of each of the two directions. The small number on the right side of the endpoint of each trip is the actual headway with respect to the previous trip, the length of the larger base of the trapezoid represents the total waiting time at the corresponding terminal, and the length of the smaller base represents the extra waiting time at the terminal with respect to the minimum required; hence, if the trapezoid reduces to a triangle, no extra waiting time was used, and therefore no secondary cost is incurred. Fig. 9 represents (a slice of) a solution where a *fully regular* (periodic) service was imposed, i.e., all the actual headways correspond to the ideal headway of 15 min, while Fig. 8 shows (the same slice of) a solution calculated by our algorithm. It is apparent from the two pictures that a fully regular service comes at a price, as the (extra) waiting times at the terminals are fairly large in Fig. 9. On the other hand, the solution in Fig. 8 is not perfectly regular (some trips have 16 min headway instead of 15), but the corresponding extra waiting times are far smaller; moreover, it uses only 8 vehicles, while the perfectly regular solution requires 9 vehicles. This shows that carefully placed minor deviations from the ideal headway, likely irrelevant to users, can lead to a much better usage of the resources (i.e., number of vehicles and waiting times). It is also clear how manually finding such a carefully crafted solution can be tedious and time consuming, even for an experienced planner, while our approach produces it automatically.

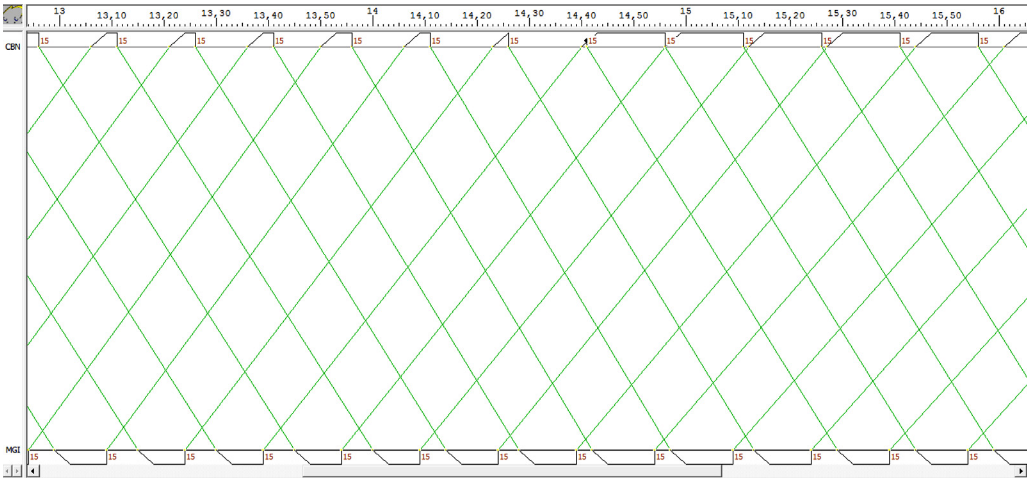

Fig. 9. Screenshot of a “completely regular” solution at peak hours.

References

Bertossi, A., Carraresi, P., Gallo, G., 1987. On some matching problems arising in vehicle scheduling models. Networks 17, 271–281.

Borghetti, A., Frangioni, A., Lacalandra, F., Nucci, C., February 2003. Lagrangian heuristics based on disaggregated bundle methods for hydrothermal unit commitment. IEEE Trans. Power Syst. 18 (1), 313–323.

Borndörfer, R., Karbstein, M., Liebchen, C., Lindner, N., 2018. A Simple Way to Compute the Number of Vehicles That Are Required to Operate a Periodic Timetable. In: Borndörfer, R., Storandt, S. (Eds.), 18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany 16:1–16:15 doi:10.4230/OASIcs.ATMOS.2018.16.

Borndörfer, R., Liebchen, C., 2008. When periodic timetables are suboptimal. In: Operations Research Proceedings 2007. Springer, pp. 449–454.

Borndörfer, R., Reuther, M., Schlechte, T., K. Waas, K., Weider, S., 2015. Integrated optimization of rolling stock rotations for intercity railways. Transp. Sci. 50 (3), 863–877.

Bunte, S., Kliewer, N., 2009. An overview on vehicle scheduling models. Public Transport 1 (4), 299–317.

Cappanera, P., Frangioni, A., 2003. Symmetric and asymmetric parallelization of a cost-Decomposition algorithm for multi-Commodity flow problems. INFORMS J. Comput. 15 (4), 369–384.

Carpaneto, G., Dell'Amico, M., Fischetti, M., Toth, P., 1989. A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks 19 (5), 531–548.

Ceder, A., 2001. Efficient Timetabling and Vehicle Scheduling for Public Transport. Springer, pp. 37–52.

Ceder, A., Wilson, N., 1986. Bus network design. Transp. Res. B 20 (4), 331–344.

Chakroborty, P., Deb, K., Sharma, R., 2001. Optimal fleet size distribution and scheduling of transit systems using genetic algorithms. Transp. Plan. Technol. 24 (3), 209–226.

The Coin-OR Linear Programming project, The Coin-OR linear programming project, https://projects.coin-or.org/clp.

Crainic, T., Frangioni, A., Gendron, B., 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design problems. Discrete Appl. Math. 112, 73–99.

Daduna, J.R., Paixão, J.M.P., 1995. Vehicle scheduling for public mass transitan overview. In: Computer-aided transit scheduling. Springer, pp. 76–90.

Desaulniers, G., Hickman, M., 2007. Public Transit. In: G. Laporte, C.B. (Ed.), Transportation. In: Handbooks in Operations Research and Management Science, 14, pp. 69–127.

Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F., 1995. Time constrained routing and scheduling. In: Bal, M., Magnanti, T., Monma, C., Nemhauser, G. (Eds.), Trends in Applied Intelligent Systems. In: Operations Research and Management Science, 8, pp. 21–30.

Fleurent, C., Lessard, R., Seguin, L., 2007. Transit timetable synchronization: evaluation and optimization. Technical Report GIRO.

Fleurent, C., Lessard, R., Seguin, L., 2009. Integrated Timetabling and Vehicle Scheduling. Technical Report GIRO.

Fonseca, J., van der Hurk, E., Roberti, R., Larsen, A., 2018. A matheuristic for transfer synchronization through integrated timetabling and vehicle scheduling. Transp. Res. Part B 109, 128–149.

Frangioni, A., 2002. Generalized bundle methods. SIAM J. Optim. 13 (1), 117–156.

Frangioni, A., 2005. About Lagrangian methods in integer optimization. Ann. Oper. Res. 139 (1), 163–193.

Frangioni, A., Gallo, G., 1999. A bundle type dual-Ascent approach to linear multicommodity min cost flow problems. INFORMS J. Comput. 11 (4), 370–393.

Frangioni, A., Gentile, C., Lacalandra, F., 2008. Solving unit commitment problems with general ramp contraints. Int. J. Electr. PowerEnergy Syst. 30, 316–326.

Frangioni, A., Gorgone, E., 2014. Generalized bundle methods for sum-Functions with “Easy” components: applications to multicommodity network design. Math. Program. 145 (1), 133–161.

Frangioni, A., Lodi, A., Rinaldi, G., 2005. New approaches for optimizing over the semimetric polytope. Math. Program. 104 (2–3), 375–388.

Guihaire, V., Hao, J., 2007. Improving timetable quality in scheduled transit networks. In: Garcia-Pedrajas, N., Herrera, F., Fyfe, C., Bendtez, J., Ali, M. (Eds.), Trends in Applied Intelligent Systems. In: Lecture notes on computer science, 6096, pp. 21–30.

Guihaire, V., Hao, J., 2010. Transit network timetabling and vehicle assignment for regulating authorities. Comput. Ind. Eng. 59 (1), 16–23.

Guihaire, V., Hao, J.-K., 2008. Transit network design and scheduling: a global review. Transp. Res. Part A 42 (10), 1251–1273.

Ibarra-Rojas, O., Giesen, R., Rios-Solis, Y., 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. Part B 70, 35–46.

Kliewer, N., Mellouli, T., Shul, L., 2005. A time-space network based exact optimization model for multi-depot bus scheduling. Eur. J. Oper. Res. 175 (1), 1616–1627.

Liu, T., Ceder, A., 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.

Liu, T., Ceder, A., Chowdhury, S., 2017. Integrated public transport timetable synchronization with vehicle scheduling. Transportmetrica A 13 (10), 932–954.

The mcfclass project The mcfclass project, http://www.di.unipi.it/optimize/software/mcf.html.

Michaelis, M., Schöbel, A., 2009. Integrating line planning, timetabling, and vehicle scheduling: a customer-oriented heuristic. Public Transport 1 (1), 211–232.

Pätzold, J., Schiewe, A., Schiewe, P., Schöbel, A., 2017. Look-Ahead Approaches for Integrated Planning in Public Transportation. In: D'Angelo, G., Dollevoet, T. (Eds.), 17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany 17:1–17:16 doi:10.4230/OASIcs.ATMOS.2017.17.

Petersen, H., Larsen, A., Madsen, O., B. Petersen, S.R., 2012. The simultaneous vehicle scheduling and passenger service problem. Transp. Sci. 47 (4), 603–616.

Rangaraj, N., Sohoni, M., Puniya, P., Garg, J., 2006. Rake linking for suburban train services. OPSEARCH 43 (2), 103–116. doi:10.1007/BF03398768.

Rostami, B., Chassein, A., Hopf, M., Frey, D., Buchheim, C., Malucelli, F., Goerig, M., 2018. The quadratic shortest path problem: complexity, approximability, and solution methods. Eur. J. Oper. Res. 268 (2), 473–485.

Schmid, V., Ehmke, J., 2015. Integrated timetabling and vehicle scheduling with balanced departure times. OR Spectrum 37 (4), 903–928.

Schoebel, A., Schiewe, A., Albert, S., Paetzold, J., Schiewe, P., Schulz, J., 2018. LINTIM : an integrated environment for mathematical publictransport optimization. Preprint-Serie des Instituts für Numerische und Angewandte MathematikLotzestr Goettingen.

Schrijver, A., 1993. Minimum circulation of railway stock. CWI Q. 6 (3), 205–217.

Serafini, P., Ukovich, W., 1989. A mathematical model for periodic scheduling problems. SIAM J. Discrete Math. 2 (4), 550–581.

Van den Heuvel, A., Van den Akker, J., Van Kooten Niekerk, M., 2008. Integrating timetabling and vehicle scheduling in public bus transportation. Technical Report UUCS2008003.

Weiszer, M., Fedorko, G., Cŭjan, Z., 2010. Multiobjective evolutionary algorithm for integrated timetable optimization with vehicle scheduling aspects. In: Perner's Contacts, 5, pp. 286–294.

Zhi-gang, L., Jin-sheng, S., 2007. Regional bus operation bi-level programming model integrating timetabling and vehicle scheduling. Syst. Eng. Theory Practice 27 (11), 135–141.