Enhancing the interaction of railway timetabling and line planning with infrastructure awareness

Florian Fuchs\textsuperscript{a}, Alessio Trivella\textsuperscript{b,∗}, Francesco Corman\textsuperscript{a}

\textsuperscript{a} Institute for Transport Planning and Systems, ETH Zürich, 8092 Zürich, Switzerland
\textsuperscript{b} Industrial Engineering and Business Information Systems, University of Twente, 7500 AE Enschede, The Netherlands

\textbf{A R T I C L E   I N F O}

Keywords:
Timetabling
Line planning
Integrated planning
Railways
Feasibility cuts

\textbf{A B S T R A C T}

Planning a railway system is done in multiple stages that are typically intractable to optimize in an integrated manner. This work develops a novel iterative approach to tackle two of these stages jointly: line planning and timetabling. Compared to existing approaches that iteratively ban a whole conflicting line plan when the timetable is found infeasible, our method can accurately identify the smallest set of incompatible services. Besides, by efficiently exploiting the available railway infrastructure, our method accounts for all the possible routing options of trains, a feature commonly neglected to reduce complexity but that helps gaining timetable feasibility. Using real data from a railway company in Switzerland, we find that our approach is (i) practical for solving real-life instances, (ii) an order of magnitude faster than existing benchmarks, and (iii) able to solve more instances. Our insights shed light on the necessity of considering infrastructure and banning conflicts rather than line plans in the joint line planning and timetabling problem.

1. Introduction

Driven mostly by population and economic growth, the demand of rail transport in Europe is expected to increase substantially in the next few decades (EC, 2017). For example, Switzerland predicts its transport demand on railways to increase by 51% for passenger and 45% for freight by 2040 compared to the 2016 levels (ARE, 2016). Given these prospects and the fact that railway systems in many areas are already at the edge of their capacity limit, the Swiss and European visions aim to utilize existing infrastructure more efficiently (Stölzle et al., 2015). The research in this paper aligns with this goal by outlining a framework for improved planning of railway services, which are able to better exploit the available infrastructure and therefore cope with some additional demand.

The typical approach to plan services in a railway system covers different time horizons and several stages (Liebchen and Möhring, 2004; Lusby et al., 2018). These stages are illustrated in Fig. 1 and range from the strategic network development involving a multi-year planning horizon, to the tactical planning stages such as timetabling and vehicle scheduling, to the real-time traffic management problem. Each stage has specific goals, requires different information and level of details, and has been formulated mathematically in the extant literature as an optimization problem.

In both the state-of-art academic research and practice, these stages are most commonly solved individually, i.e., independently one from another. Clearly, splitting the overall planning process over multiple stages has the advantage that individual tasks are less challenging to solve (although for complex railway systems even solving one of them may be daunting). Nevertheless, since the solution of an intermediate stage depends on the input from the previous stages, it is well known that approaches considering...
multiple stages jointly have the potential to improve the overall planning solution. As the pressure to better exploit the available infrastructure grows, such approaches are increasingly demanded by practitioners (Schiewe, 2020).

The current paper tackles two stages that are traditionally solved at different time horizons and cover strategic and tactical decisions, namely the line planning problem (LPP) and the timetabling problem (TTP). We also include in our focus constraints from the infrastructure by considering the routing and platforming problem. The latter problem is illustrated as part of the timetabling stage in Fig. 1 but is usually ignored while determining a timetable. In effect, it could be considered as a third planning stage that we address jointly with the LPP and TTP. Finally, we neglect network planning as we aim to optimize existing networks, as well as vehicle and crew scheduling that are often addressed on shorter-term horizons compared to the LPP and TTP (Lusby et al., 2018).

The joint consideration of the LPP and TTP is not new in the literature and has been addressed with a sequential, iterative, or integrated approach. Sequential approaches strictly separate the two tasks by finding a line plan before computing the timetable. Integrated approaches tackle both problems in a single model, likely providing superior solutions. However, such models are intractable to solve for real-life instances (Schiewe, 2020). Instead, iterative approaches go back and forth between the two problems and provide a balance between complexity and solution quality (Schöbel, 2017).

Existing iterative approaches have only explored the feedback loop between timetabling and line planning in a simplified manner when facing a conflict (i.e., an infeasibility) during timetabling. Specifically, these approaches work by adapting the line which has the lowest buffer times (Burggraeve et al., 2017), restricting the frequencies of all lines (Yan and Goverde, 2019), considering only station capacity (van Lieshout et al., 2020), or completely ignoring infrastructure capacity (Fuchs and Corman, 2019). In the current paper, we propose an iterative approach to solve the joint LPP and TTP, which we enhance by designing a tailored procedure exploiting feasibility cuts. Compared to the state-of-the-art iterative methods, we address in a novel manner the problem of how to best provide feedback (i.e., banning some solutions) from an infeasible or poorly performing timetable instance (a conflict) derived from the solution to a line planning problem.

Furthermore, existing line planning and timetabling approaches do not consider the available itineraries for the trains, i.e., they are not aware of the detailed infrastructure limitations when determining solutions or conflicts. In fact, the choice of different routes and platforms at stations (termed routing and platforming) is not considered at a detailed level in an integrated manner and is often performed as a successive follow-up of the timetabling problem (Caprara et al., 2011; Zhang et al., 2020). Our approach is instead oriented to infrastructure awareness while generating feedback. This allows us to exploit at best the scarce infrastructure capacity and avoid overestimating it (assuming it is infinite) or underestimating it (and effectively waste the potential from the available resources).

To summarize, we address these two gaps in the literature with the following contributions:

1. We outline a novel iterative approach for the joint LPP and TTP that accurately identifies and bans the set of conflicting services in the LPP to find conflict-free solutions to the associated TTP.
2. We provide a domain model and problem formulation able to exploit the available infrastructure by enabling train itinerary assignment during timetabling (i.e., routing and platforming).
3. We theoretically assess the benefit of banning conflicts compared to banning line plans, and show the implications of this towards convergence speed and generated cuts.
4. We show the value and suitability of the proposed approach by means of an extensive numerical study using a large real-life instance from a railway company in Switzerland.

The paper is structured as follows. In Section 2, a literature review provides the necessary background material. In Section 3, we overview our iterative approach and examine its advantages theoretically. In Section 4, we discuss our methodology to tackle the LPP, the TTP, and the feedback between them. We present a computational study in Section 5, providing both algorithmic and practical insights, and conclude in Section 6.
2. Literature review

We review the literature on LPP in Section 2.1, TTP in Section 2.2 and the joint consideration of them in Section 2.3.

2.1. Line planning problem

In the scope of this paper, a line represents a set of train services that visit (stopping or not) the same sequence of stations in regularly-spaced time intervals. Therefore, as indicated by the name, it is the main task of the LPP to derive a line plan, i.e., a set of lines with specified frequencies and rolling stock, to route passengers in the most efficient way. Line plans aim to satisfy passenger demand; thus, minimizing the total travel time is the commonly considered objective. As the literature on the LPP is vast, we refer to Schöbel (2012) and Schmidt and Schöbel (2015a) for an overview of different models and methods, and only highlights a few key works here.

By using the public transport network to route passengers along lines, Schöbel and Scholl (2006) derive a mixed-integer program (MIP) for the LPP. Goerigk and Schmidt (2017) enhance the travel time minimization approach by enforcing passengers to be routed along the shortest path available. Operating a line plan also generates costs that may differ significantly depending on the chosen line plan. These costs can be integrated in the LPP with a budget constraint as shown by Friedrich et al. (2017). Moreover, as the required rolling stock highly affects the operating costs, Goossens et al. (2006) propose to account for the rolling stock by incorporating constraints at the LPP stage.

Most of the published approaches use a pool of candidate lines as input to the LPP (Schöbel, 2012), as we also do in this paper. Gattermann et al. (2017) show instead how to generate a line pool algorithmically. It is worth mentioning that the choice of a line pool can affect the resulting line plan, timetable, and rolling stock roster (Pätzold et al., 2017). In contrast to the above approaches, Borndörfer et al. (2007) propose a column generation method that does not require any line pool as input and maximizes the number of direct travelers but does not yield integer line frequencies.

Overall, the LPP is well studied in the transport and operations research literature. Thereby, considering it jointly with other planning stages is a reasonable research focus (Schiewe, 2020).

2.2. Periodic timetabling problem

The periodic TTP in railways (henceforth simply TTP) is usually modeled as an adaptation or generalization of the periodic event scheduling problem (PESP), first proposed by Serafini and Ukovich (1989). Similarly to the LPP, the literature on PESP/TTP is vast and we do not attempt to review it here in full. We instead refer to Liebchen and Möhring (2004) and Caimi et al. (2017) for an overview and applications of the TTP, and to Peeters (2003) for detailed insights on cycle-based formulations.

The TTP aims to assign times to activities (e.g., the arrival and departure of trains at stations) subject to specific railway constraints (e.g., headways). The objective can be to reduce the total travel time or the number of required vehicles (Liebchen and Möhring, 2004). Since travel times often depend on the passenger’s routes, choosing such routes might be beneficial as shown by Gattermann et al. (2016). An analogous finding is given by Robenek et al. (2016) for non-periodic timetables.

Besides periodicity, a timetable might also be required to feature symmetry (i.e., equal transfer times for each inbound and outbound trip), which imposes further constraints to a TTP model and could worsen the quality of the solution as shown by Liebchen (2004). The TTP has also been adapted to allow track allocation. In particular, the approach suggested by Wüst et al. (2019) is able to allocate station tracks during the solution process. Intending to insert additional trains in a timetable, Großmann (2016) enhances a previous model (Großmann, 2011).

Concerning solution methods for the PESP/TTP, MIP formulations are perhaps the most common (Caimi et al., 2017). Besides, the approach by Großmann (2011) to address the PESP as a Boolean satisfiability problem (SAT) has been successfully applied to real-life railway instances (Kümmling et al., 2015). Herrigel et al. (2018) use the cycle-based MIP to iteratively build a timetable from scratch. Given an existing timetable, Goerigk and Liebchen (2017) outline a refined algorithm to improve the objective. A different approach based on dynamic programming using time–space graphs is presented by Zhang et al. (2019). Recently, Borndörfer et al. (2020) proposed a concurrent approach combining SAT and MIP with improving algorithms to provide better solutions for benchmark instances. We will consider both SAT and MIP formulations in this paper.

As the TTP is well researched, several problem formulations and solution methods have been developed in the literature (Schiewe, 2020). However, large-scale instances still impose a challenge. Furthermore, only a few approaches account for the routing and platforming of vehicles within the TTP.

2.3. Joint consideration of line planning and timetabling

According to the extant literature and the extensive assessment by Schiewe (2020), the three typical strategies to integrate multiple planning stages are shown in Table 1. This also applies to the joint solution of the LPP and TTP. In particular, although both tasks are already challenging, considering them simultaneously embeds further potential for improving solution quality.

One of the few examples of an integrated approach is the MIP model by Lübbecke et al. (2018) to optimally solve the joint line planning, timetabling, and vehicle scheduling. Since exact approaches for integrated problems scale poorly with the instance size, iterative approaches are more common and can be seen as heuristic methods that decompose the integrated problem by stage.
Iterative methods include the Eigenmodel framework by Schöbel (2017) that allows to formally decompose stages by iteration, and the method by van Lieshout et al. (2020) to find alternative line plans in out-of-control situations by decomposing the integrated problem into a master and slave subproblems. While the master problem is an LPP that considers passengers and vehicles, the slave problem solves a TTP for each station. In case of insufficient capacity at stations, the conflicting lines are added as a cut to the master problem. By rearranging the traditional sequence of line planning, timetabling, and vehicle scheduling, Pätzold et al. (2017) propose a look ahead heuristic that restricts the set of candidate lines to the ones that use their vehicles efficiently. In fact, accounting for the vehicle schedule at such an early stage yields timetables that are more efficient to operate. Fuchs and Corman (2019) focus on a similar aspect by developing an iterative method that considers the scheduling of vehicles in all subproblems. By incorporating flexible connections and vehicle rotations, the resulting timetable is optimized from the perspective of both passengers and operator. Similarly, Michaelis and Schöbel (2009) account for customers from the start when considering a line planning, timetabling, and vehicle scheduling problem. Yan and Goverde (2019) combine line planning and timetabling but only consider direct travelers and iterate between the stages, restricting the maximal frequencies until reaching a feasible and acceptable solution. Finally, Burggraeve et al. (2017) consider line planning and timetabling using an iterative approach based on resolving insufficiently performing timetables by banning the corresponding line planning solution and adapting the most critical line. The most critical line is the line which has the lowest buffer times in the resulting timetable.

Despite several iterative methods have been developed in the literature, a central issue neglected so far is that the LPP and TTP rely on infrastructural data at different resolutions, that is, a line plan does not guarantee a feasible timetable. The existing works overcome this issue by retrieving the second-best line plan (Burggraeve et al., 2017), restricting the frequencies of all lines based on assumptions (Yan and Goverde, 2019), relaxing some constraints (Kümmling et al., 2015), or assuming that infrastructure capacity can be completely ignored (Fuchs and Corman, 2019). The approach of van Lieshout et al. (2020) is the only one to our knowledge that explicitly addresses conflicts due to insufficient infrastructure availability. However, as this approach only assesses station capacity, global timetable feasibility is not ensured.

Given the simplified techniques used so far to address the feedback loop from TTP to LPP (which we see as a gap in the literature), our approach distinguishes itself by explicitly deriving conflicts or bottlenecks when the TTP is infeasible. Thereby, we enhance the state of the art of iterative approaches by addressing conflicts explicitly instead of solely relying on assumptions for approximately integrating infrastructure conflicts. Especially in large and busy networks, it is essential to provide the LPP with accurate constraints; as otherwise, two scenarios may arise. Firstly, in the case that too conservative and restrictive assumptions are used, the iterative procedure may fail to fully exploit the available infrastructure, leading to inferior solutions. Secondly, in case of too loose constraints, the performance of the iterative approach degrades as the LPP solution is increasingly likely to be infeasible in the subsequent planning stage, causing many iterations and/or loss of convergence. In particular, we show the effectiveness of our approach by comparing it theoretically and numerically with a simple, iterative approach. The simple approach lacks infrastructure information and iteratively bans a line planning solution rather than addressing the conflict.

An additional challenge arises when generating a TTP instance related to an LPP solution. In practice, different routing options for trains might be available; hence, it is essential to perform the train routing and platforming based on a detailed representation of the infrastructure as well as its concurrent usage by multiple services in the line plan (Caimi et al., 2017; Liao et al., 2021). For this reason, the TTP model we develop accounts for all available routing options of services, which allow to better exploit the underlying infrastructure and consequently better identify conflicts.

3. Approach and theoretical motivation of the feedback scheme

In this section, we first outline the structure of our iterative approach in Section 3.1 and introduce some essential definitions and notation in Section 3.2. Then, we motivate this approach by discussing an illustrative example in Section 3.3. Finally, we present more general theoretical results on the convergence of the method when banning conflicts versus line plans in Section 3.4.

### 3.1. Setup of the iterative approach

As shown in Fig. 2, our approach consists of an LPP module, a TTP module, and an interface module connecting them.

The LPP module computes optimal line plans that minimize the total travel time to route passengers in the railway network while accounting for the available vehicles. We tackle this problem using a state-of-the-art MIP formulation that was introduced by Schöbel and Scholl (2006), refined in more recent works (Schmidt and Schöbel, 2015b; Goerigk and Schmidt, 2017; Burggraeve et al., 2017), and adapted here to our iterative approach that bans conflicts.

Given an input line plan, the TTP module generates periodic timetables based on the formulation of Serafini and Ukovich (1989), later adapted for various applications (Caimi et al., 2017). Specifically, we propose a TTP formulation that is aware of the available

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Solution quality</th>
<th>Computational effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated</td>
<td>Optimal</td>
<td>Intractable</td>
</tr>
<tr>
<td>Iterative</td>
<td>Good</td>
<td>High</td>
</tr>
<tr>
<td>Sequential</td>
<td>Poor</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 1: Strategies to account for multiple planning stages. Source: Adapted from Schiewe (2020).
infrastructure as it includes the routing and platforming of trains. Since our focus regarding the TTP is mostly on feasibility rather than on optimality, we additionally encode the TTP problem as an SAT, which can be more efficient than MIP formulations in establishing timetable feasibility (Borndörfer et al., 2020). The SAT encoding is also key in our approach to detect the smallest set of conflicting services in a line plan.

Using the interface module, the LPP and TTP modules interact in both directions. First, the LPP is solved and its solution converted to an input for the TTP. Then, the TTP is solved, and, if it is infeasible, the set of conflicting services is extracted. Based on this set, we generate a cut in the LPP model such that the line planning solutions retrieved in the following iterations will no longer suffer from the identified conflict. When the TTP is found feasible, an optimized solution can be determined, and the process ends.

The reason why we focus on timetable feasibility during the iterative approach and only optimize the final (feasible) timetable is the following. In general, we cannot ensure that the joint LPP-TTP solution generated with our method is optimal concerning an integrated problem whose objective function combines that of the LPP and the TTP. In our case, however, we employ an objective function that is consistent across LPP and TTP. Indeed, both mathematical programming formulations we propose in Section 4 minimize the total travel time to route all passengers from their origin to destination. Thus, although these objectives are not exactly the same (due to the different granularity of LPP and TTP and decision variables used), the intuition is that they are highly correlated and that LPP solutions with low total travel time lead to TTP solutions with also low total travel time. Thus, focusing on timetable feasibility (using a SAT solver) and potentially optimizing the final timetable seems sufficient when employing consistent objectives. Incorporating different TTP objectives is certainly possible when optimizing the final timetable. However, it would be harder to guess how good the joint LPP-TTP solution would be from the perspective of an integrated problem and objective function.

3.2. Essential notation and definitions

In this section, we introduce the main notation used throughout the paper, which is not meant to be comprehensive since additional notation will be provided at a later stage as needed. The entire set of parameters, symbols, and variables used in the different models is described in Appendix A.

We consider a time interval of operations denoted $T$ (e.g., one hour). Passenger demands are given as a set of origin–demand pairs $od \in \mathcal{OD}$, where $o$ is the origin station and $d$ the destination. Each $od$ is associated with the amount of demand $p_{od}$ between $o$ and $d$ for the considered period $T$.

The line pool $X_{\text{pool}}$ is the set of candidates used as input for the LPP. An element of this set is identified by a triple $(l, f, r)$, where $l$ represents the line (identified by the sequence of visited stations where the service stops or not), $f$ its periodical frequency within $T$, and $r$ the assigned rolling stock configuration. We call $\mathcal{L}$ the set of lines, $\mathcal{F}_i$ the set of available frequencies for a line $i \in \mathcal{L}$, and $\mathcal{R}_r$ the set of possible vehicle compositions. Thus, $X_{\text{pool}} = \mathcal{L} \times \mathcal{F}_i \times \mathcal{R}_r$. Note that the set of lines $\mathcal{L}$ alone differs from $X_{\text{pool}}$ whose elements combine lines, frequencies, and vehicles.

A line plan is a solution to the LPP and is a combination of elements $(l, f, r) \in X_{\text{pool}}$ that jointly fulfill the passenger demand relations in $\mathcal{OD}$; it is denoted by $X_{\text{plan}} \subseteq X_{\text{pool}}$. The line plan can equivalently be identified by means of a vector of binary indicators $x_{l,f,r} \in \{0,1\}$ that are set to one when the candidate $(l, f, r) \in X_{\text{pool}}$ is selected in the line plan, and set to zero otherwise.

Finally, we elaborate on vehicle compositions $r \in \mathcal{R}_r$. Compositions in the set $\mathcal{R}_r$ are made of a different number and types of basic rolling stock units (e.g., regular carriage, freight carriage, auto-train carriage, engine, etc.). Therefore, we need to introduce a rolling stock unit type $g \in \mathcal{G}$, which is available in limited number $g_{\text{max}}$. To track the underlying structure of each composition $r \in \mathcal{R}_r$, we use a parameter $\text{mix}_{g,r}$ that specifies the number of units of type $g$ used in a train composition $r$. When $r \in \mathcal{R}_r$ is chosen, it provides an activated candidate $(l, f, r)$ with a capacity $cap_r$.
3.3. Illustrative example

To illustrate the benefit of the new feedback scheme proposed, we constructed a small instance comprising of five stations and shown in Fig. 3 along with infrastructure, demand, and line pool.

As shown in the central panel of this figure, OD includes three passenger demand relations to fulfill: $\gamma^{1-5} = 1$, $\gamma^{2-5} = 2$, and $\gamma^{3-5} = 3$ demand units. For simplicity, in this example we restrict the line pool by considering only one direction and vehicle configuration per line. More specifically, we assume that the line frequency is always equal to two ($f = 2$), and that $r$ provides sufficient capacity for all passengers and sufficient vehicles are available to operate all possible permutations of candidate configurations. This leads to $|X_{pool}| = 6$ as shown in the right panel of Fig. 3. Finally, solving the LPP requires specifying the duration of different activities. To this end, we assume that a train covers a section between any pair of stations in 4 min, all dwells have a duration of 1 min, and all access/boarding times are equal to 5 min. Transfers are possible for any pair of lines at station 4. The minimum duration of a transfer is fixed to 5 min.

Given these specifications, we can solve the LPP by routing each demand $\gamma^{\text{od}}$ via the shortest path through a network made of the six available lines possibly interconnected by means of transfers. There are seven feasible line plans that we report in Table 2 ranked by objective function, i.e., total travel time. We omit displaying the configurations where at least one element $(l, f, r)$ is activated but is not used by any passenger (these redundant configurations have an objective function value equal to one of the already listed solutions). The seven feasible line plans are visualized in Fig. 4 and constitute each a valid input to the TTP. Transfers are identified by dotted arcs between arrivals and departures of different lines and do not affect the feasibility of the line plans.

In case no conflict exists among the elements of these line plans, the best performing LPP solution (i.e., $S-1$) leads to a feasible timetable and no further interaction between $LPP$ S-3, S-4, in this order, to eventually determine a feasible timetable once the LPP provides S-5 as a solution. Thus, 5 iterations are needed to produce a feasible timetable, i.e., the LPP and TTP have to be solved 5 times each. If we instead look at the conflicts, three conflicts exists: $X^{1}_{\text{conflict}} = \{(l_1, 2, r), (l_2, 2, r)\}$, $X^{2}_{\text{conflict}} = \{(l_1, 2, r), (l_1, 2, r), \}$, and $X^{3}_{\text{conflict}} = \{(l_2, 2, r), (l_1, 2, r)\}$. Thus, $|C| = 3$. Assuming that a conflict can be detected after solving the TPP, an approach that extracts conflicts would cut off S-2, S-3, S-4 (in any order), before the LPP returns S-5 admitting a feasible timetable, which makes 4 iterations in total when counting the first iteration giving S-1. Thus, detecting and extracting conflicts rather than banning a line plan makes the approach converge in one less iteration in this example.

Notice that for this simple case, it would have been easy to include upfront a restriction $x_{l_1, 2,r} + x_{l_1, 2,r} + x_{l_1, 2,r} \leq 1$ to the LPP. This constraint limits the number of active lines between stations 4 and 5, thereby banning all conflicting LPP solutions. However, deriving such constraints is generally a complex task that requires investigating timetable feasibility, as conflicts depend on speed profiles, stopping patterns, and the network structure. We discuss how we tackle this task efficiently in Section 4.

![Fig. 3. Input data to the illustrative example (see web version for interpretation of colors).](image-url)

Table 2

<table>
<thead>
<tr>
<th>Line plan</th>
<th>Configuration</th>
<th>Travel time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_{l,2,r}$</td>
<td>$t_{p,s}$, $t_{p,r}$</td>
</tr>
<tr>
<td>S-1</td>
<td>1 1 1 0 0 0 14 28 42 84</td>
<td></td>
</tr>
<tr>
<td>S-2</td>
<td>0 1 1 1 0 0 18 28 42 88</td>
<td></td>
</tr>
<tr>
<td>S-3</td>
<td>1 0 1 1 1 0 14 36 42 92</td>
<td></td>
</tr>
<tr>
<td>S-4</td>
<td>1 1 1 0 1 0 14 28 54 96</td>
<td></td>
</tr>
<tr>
<td>S-5</td>
<td>0 0 0 1 1 0 18 36 42 96</td>
<td></td>
</tr>
<tr>
<td>S-6</td>
<td>0 1 0 1 0 1 18 28 54 100</td>
<td></td>
</tr>
<tr>
<td>S-7</td>
<td>1 0 0 0 1 1 14 36 54 104</td>
<td></td>
</tr>
</tbody>
</table>
3.4. Motivation for banning conflicts versus line plans

Motivated by the previous example, we theoretically assess the benefit of banning an $\mathcal{C}$ conflict instead of a $\mathcal{L}$ plan more in general. We will first focus on one step of the iterative approach, and then study the impact on the overall number of required iterations. The following lemma states a simple relation on the cardinality of a conflict set, a line plan, and a line pool.

**Lemma 1.** For any conflict $\mathcal{C}$ identified in a line plan $\mathcal{L}$, it holds that

$$0 < |\mathcal{C}| \leq |\mathcal{L}| \leq |\mathcal{P}|.$$  

**Proof.** By definition of line pool, any line plan $\mathcal{L}$ may only contain elements $(l, f, r)$ from this pool, i.e., $\mathcal{L} \subseteq \mathcal{P}$. Furthermore, any $\mathcal{C}$ may only contain elements $(l, f, r)$ that are part of $\mathcal{L}$, as otherwise not all $(l, f, r) \in \mathcal{C}$ are in $\mathcal{L}$, eliminating the conflict in first place. This means that $\mathcal{C} \subseteq \mathcal{L}$ holds for any line plan that suffers from a conflict. Finally, a conflict $\mathcal{C}$ must include at least one candidate $(l, f, r)$, and the same must hold for any line plan $\mathcal{L}$, as otherwise the absence of routing options makes the line plan clearly infeasible. □

Next, let us define $|\mathcal{P}| = \alpha$. Proposition 1 shows that banning a conflict instead of a line plan removes in best case an exponential number of distinct and infeasible line plans simultaneously.

**Proposition 1.** Banning $\mathcal{C}$ removes $n$ many line plans, where $n \in \{1, \ldots, 2^{\alpha-1}\}$.

**Proof.** From (1) we know that $|\mathcal{C}| \in \{1, \ldots, |\mathcal{L}|^*\}$, where $\mathcal{L}^*$ is the line plan from which the conflict is extracted. Consequently, two extreme cases exist: $|\mathcal{C}| = 1$ and $|\mathcal{C}| = |\mathcal{L}|^*$. Without loss of generality we can assume that any $\mathcal{L} \subseteq \mathcal{P}$ is a valid LPP solution, meaning that $2^\alpha - 1$ distinct $\mathcal{L}$ can be found. If we now ban an $\mathcal{C}$ with $|\mathcal{C}| = 1$, we are effectively removing one element $(l, f, r)$ from $\mathcal{P}$, which corresponds to decreasing the cardinality of the line pool $\alpha$ by 1, i.e., the number of possible LPP solutions would reduce to $2^{\alpha-1} - 1$. Thereby, banning an $\mathcal{C}$ where $|\mathcal{C}| = 1$ will cut off $2^{\alpha-1}$ distinct $\mathcal{L}$ from the LPP. At the opposite extreme, if $\mathcal{C} \equiv \mathcal{L}^*$ and $\mathcal{L}^* \equiv \mathcal{P}$, then banning $\mathcal{C}$ is equivalent to banning $\mathcal{L}^*$ and thus removes exactly one line plan $(\mathcal{L}^*)$ from the LPP. □

Proposition 2 assesses the impact of banning conflicts rather than line plans on the convergence of the iterative approach, i.e., on the number of iterations needed to obtain a feasible timetable.

![Fig. 4. Visualization of the feasible line plans (see web version for interpretation of colors).](image-url)
Proposition 2. Instances exist such that a method iteratively banning $X_{\text{conflict}}$ converges in $\mathcal{O}(\alpha)$ iterations, whereas a method iteratively banning $X_{\text{plan}}$ requires $\mathcal{O}(2^\alpha)$ iterations.

Proof. The proof of this proposition is based on constructing an instance that fulfills the statement. To this purpose, we generalize the example in Section 3.3 to $\alpha$ stations $\{1, \ldots, \alpha\}$ (instead of just $\{1, 2, 3\}$) connected to a central station $\alpha + 1$ (previously station 4) and an additional station $\alpha + 2$ (previous station 5). The demand from station $j \in \{1, \ldots, \alpha\}$ to station $\alpha + 2$ is equal to $j$. Similar to the example in Section 3.3, we have main lines $l_j$ connecting station $j$, $\alpha + 1$, and $\alpha + 2$, and feeder lines $l_{j\alpha}$ only connecting $j$ with $\alpha + 1$, but we assume here that all feeder lines are mandatory, i.e., always active. Thus, the task of the LPP is to select among the main lines. We use the same assumption as Section 3.3 regarding frequency and vehicles, which implies that $|X_{\text{pool}}| = \alpha$. We also assume that track capacity between stations $\alpha + 1$ and $\alpha + 2$ is scarce and only the first element $(l_j, r)$ can be operated, i.e., any $(l_j, r)$ for $j > 1$ would consume too much capacity and make the TTP infeasible.

Let us examine what happens when banning line plans in this instance. The LPP admits $2^\alpha - 1$ feasible solutions, which are all non-empty subsets of $X_{\text{pool}}$, but only one of them is feasible to the TTP, i.e., the singleton $X_{\text{plan}} = \{(l_1, 2, r)\}$. When solving the LPP, solutions using multiple lines will be initially favored due to shorter connections, and banned one after the other. Among the LPP solutions using a single line, it is easy to verify that $\{(l_1, 2, r)\}$ will be chosen last as it yields the lowest objective function score due to the lowest passenger demand at station 1. In conclusion, we inevitably have to check all possible $2^\alpha - 1$ combinations before getting to $X_{\text{plan}} = \{(l_1, 2, r)\}$ and hence a feasible timetable.

If we instead consider conflicts, we have $\alpha - 1$ of them, which are made of singletons $X_{\text{conflict}} = \{(l_2, 2, r)\}$ to $X_{\text{conflict}} = \{(l_\alpha, 2, r)\}$. Therefore, banning one conflict per iteration only requires $\alpha$ iterations before retrieving $X_{\text{plan}} = \{(x_j, 2, r)\}$ and the corresponding feasible timetable. □

In this section, we have shown that banning conflicts never requires more iterations compared to banning line plans. There exist even cases where the former strategy converges in a linear number of iterations in $|X_{\text{pool}}|$ whereas the latter requires an exponential number of iterations. The proofs of Propositions 1–2 suggest that banning conflict sets that are small in size, i.e., $|X_{\text{conflict}}| \ll |X_{\text{plan}}|$ is particularly beneficial to speed up convergence since more invalid line plans are cut off per iteration. For large real-life networks, we expect that conflicts only involve small subsets of all lines used and that the advantage of our approach is therefore substantial. We will show this is indeed the case in our numerical study in Section 5.

4. Enhanced iterative methodology

In this section, we discuss in detail our iterative approach. We introduce our domain model in Section 4.1, focus on the LPP module and techniques to speed up execution in Section 4.2, and eventually tackle in Section 4.3 the TTP module by providing a formulation to allow routing and platforming and to detect conflicts.

4.1. Domain modeling

The LPP usually considers the railway infrastructure on a coarser level compared to the TTP, meaning that a service may be routed through more than one possible itinerary during the TTP but this is not visible at the LPP stage. The domain model presented in this section is a feature that distinguishes our approach from previous works, as it accounts for varying levels of granularity while maintaining consistency across the LPP and TTP and providing an interface between the two tasks. We describe in the following the data structures employed in our approach based on lines and infrastructure specification, and relegate to Appendix B the pseudocode of the algorithm used to compute them.

Recall that a line $l \in \mathcal{L}$ represents a set of services that during the interval $T$ periodically serve the same sequence of stations. A line possesses two properties: a speed level and a flag stating whether it must be operated or is instead optional. Moreover, we associate each line $l$ with two routes, where the first route defines the sequence of stations served by the line from one terminus to the other in one direction, and the second route covers the opposite direction. Such routes apply to all $(l, f, r)$ corresponding to line $l$. We characterize a route with three data structures, or graphs, that we call the route constraint ($RC$), the route section ($RS$), and the route itinerary ($RI$), defined below.

- The $RC$ describes the sequence of visited stations in the infrastructure network. Note that both nodes with and without a planned stop are included in the $RC$. Furthermore, each $RC$ is defined between two termini, and only simple, direct paths connect two nodes.
- The $RS$ consists of all stations where the line stops to serve passengers. For each planned stop, the $RS$ contains an arrival and a departure node linked to each other in the direction of travel. Thereby, the $RS$ is particularly useful during the LPP as it provides the connections used by passengers to travel from their origin to destination.
- The $RI$ describes all available track assignments and is especially useful for the TTP. Formally, the $RI$ can be defined as a directed acyclic graph (DAG).

Fig. 5 shows an example made of 4 stations $\{A, B, C, D\}$ with a train stopping in $B$ and $D$. The bottom plot reports a possible microscopic infrastructure associated with this example. By accounting for these three structures and their interaction, our model is aware of the available infrastructure while routing passengers and assigning itineraries as we explain in detail in sections 4.2–4.3.
4.2. Solving the line planning problem

We tackle the LPP using an MIP model that, although based on existing formulations, is designed for our iterative approach to explicitly incorporate constraints that ban the identified conflicts (Section 4.2.1). We also introduce two heuristic simplifications to reduce the computing time per iteration (Section 4.2.2).

4.2.1. Formulation

The goal of the LPP is to find a line plan \( \mathcal{X}_{\text{plan}} \subseteq \mathcal{X}_{\text{pool}} \) that minimizes the total travel time while fulfilling all demand relations in \( \mathcal{D} \).

To route the \( od \in \mathcal{D} \), we combine the route sections \( RS \) for all routes (i.e., lines and directions) to create a section graph \( SG = (V, S) \). Specifically, for each station, we introduce artificial access, exit, and change nodes to model respectively boarding, alighting and transfer activities. These nodes are linked with the arrival and departure nodes of all lines stopping at that station by means of edges also termed access, exit, and change. We call the set of access and change edges jointly as the wait edges and denote them by \( S_{\text{wait}} \subseteq S \). To account for the relationship between waiting times and line frequency, we follow Burggraeve et al. (2017) and Yan and Goverde (2019) by adding a different wait edge for each possible frequency \( f \) of a line \( l \). Fig. 6 illustrates these concepts by showing the \( SG \) obtained from two lines serving three stations \( A, B, \) and \( C \). In this example, the top line (dotted) and bottom line (solid) have frequency \( f = 2 \) and \( f = 1 \), respectively, which is reflected in the number of parallel access and change edges at the stations.

All edges \( s \in S \) are directed and have an associated time duration \( t_s \) computed according to (2). In particular, the wait edges are provided with a frequency-dependent duration describing the expected waiting time. For all remaining edges, we use the minimal required traversal time scaled by a \( k_{\text{slack}} \geq 1 \) to account for technical reserves and to leave slack for some adjustments during the TTP.

\[
t_s := \begin{cases} 
\frac{T}{2 \cdot f} & \text{if } s \in S_{\text{wait}}, \\
 t_{s, \text{speed}} \cdot k_{\text{slack}} & \text{else}.
\end{cases}
\]  
(2)

To formulate the LPP as an MIP, we largely follow Burggraeve et al. (2017) and Pätzold et al. (2017) and introduce two sets of decision variables. The selection of line pool candidates \( (l, f, r) \in \mathcal{X}_{\text{pool}} \) is modeled with the binary variables \( x_{l,f,r} \). Then, for each edge \( s \in S \) and relation \( od \in \mathcal{D} \), the continuous flow variable \( p_{od}^s \) describes the amount of demand for \( od \) flowing through \( s \). In the

---

Fig. 5. Example of route with \( RC, RS, \) and \( RI \).

Fig. 6. Example of section graph with two lines and three stations.
following, $\delta^+(v) \subseteq S$ and $\delta^-(v) \subseteq S$ represent the sets of outgoing and incoming edges, respectively, to a given node $v \in V$. Moreover, the index $i \in I$ denotes a section of the railway track between two stations or other infrastructural elements (e.g., junctions) at the route constraint level of detail, where $I$ represents the full network. We indicate with $X_i \subseteq Y_i$ the subset of line pool candidates that include the section of the railway track $i$ in their route constraint. The MIP model is formulated in (3).

\[
\begin{align*}
\text{min} & \quad \sum_{od \in OD} \sum_{e \in S} f_{e} \cdot p_{e}^{od} \\
\text{subject to:} & \quad \sum_{f \in F_i} \sum_{r \in R_i} x_{i,f,r} \begin{cases} 1 & \text{if } i \text{ is mandatory,} \\
\leq 1 & \text{otherwise,} \end{cases} \quad \forall i \in L, \tag{3a} \\
& \quad \sum_{s \in \delta^+(v)} p_{s}^{od} - \sum_{a \in \delta^-(v)} p_{a}^{od} = \begin{cases} -y_{od} & \text{if } v = o, \\
y_{od} & \text{if } v = d, \quad \forall v \in V, \forall od \in OD, \tag{3b} \\
0 & \text{otherwise,} \end{cases} \\
& \quad \sum_{o \in OD} p_{s}^{od} \leq \sum_{(l,f,r) \in X_i} x_{i,f,r} \cdot cap_{f} \cdot f, \quad \forall i \in I, \tag{3c} \\
& \quad \sum_{(l,f,r) \in X_{pool}} x_{i,f,r} \cdot mix_{g,f} \cdot f \leq s_{\text{max}}, \quad \forall g \in G, \tag{3d} \\
& \quad \sum_{(l,f,r) \in X_{conflict}} x_{i,f,r} \leq |X_{\text{conflict}}| - 1, \quad \forall X_{\text{conflict}} \in C, \tag{3e} \\
& \quad \sum_{(l,f,r) \in X_{pool}} x_{i,f,r} \leq |X_{\text{conflict}}| - 1, \quad \forall X_{\text{conflict}} \in C, \tag{3f} \\
& \quad \text{variables: } x_{i,f,r} \in \{0, 1\}, \quad \forall (l,f,r) \in X_{\text{pool}}. \tag{3g} \\
& \quad p_{e}^{od} \geq 0, \quad \forall s \in S, \forall od \in OD. \tag{3i}
\end{align*}
\]

The objective function (3a) minimizes the total travel time to route all passenger relations in $OD$ through the $SG$. Constraints (3b) ensure that at most one configuration of frequencies and vehicle compositions is chosen for each line $l \in L$. This must be exactly one in case of services that are mandatory to operate, which can be the case e.g. of a freight service. Constraints (3c) enforce the conservation of passenger flows. In particular, for each $od \in OD$, the difference between inflow and outflow must be zero at nodes $v \in V \setminus \{o,d\}$, and must equal $-y_{od}$ and $y_{od}$ at origin and destination node, respectively. Constraints (3d) ensure that selected candidates do not transport more passengers than the capacity given by their vehicle composition. Moreover, (3e) limits the total amount of rolling stock units ($g \in G$) employed in the various vehicle compositions ($r \in R_i$) by $s_{\text{max}}$.

The LPP does not necessarily consider the available infrastructure at a granular level, e.g., it is not aware of the number of platforms at stations or tracks between stations (that are not captured by the $SG$ nor the index set $I$). Thus, an LPP model that minimizes total travel time would favor services running with higher frequencies, hence creating many conflicts in the TTP. To mitigate this issue, (3f) imposes an upper bound on total frequencies $f_{\text{max}}$ at each section of the railway track $i \in I$, as already successfully applied by Schmidt and Schöbel (2015a). Note that choosing the appropriate $f_{\text{max}}$ is nontrivial because too low values lead to wasted capacity and the loss of high-quality solutions, while too high values lead to many conflicts in the TTP affecting the performance of the overall iterative approach.

Finally, constraints (3g) are critical for the interaction between the TTP to the LPP as they ban the conflicts. Specifically, these constraints prevent future LPP executions from returning any of the already identified conflicts $X_{\text{conflict}} \subseteq C$ provided by TTP, by cutting off all the line plan solutions that include these conflicts. The structure of these constraints has been used by van Lieshout et al. (2020) and Burggraeve et al. (2017). In contrast to Burggraeve et al. (2017), we aim to use smallest set of conflicting services in order to prevent the same conflicts from reoccurring in the future. For each identified $X_{\text{conflict}}$, we add to $C$ (hence ban) all configurations that use the same lines $l \in L$ and vehicle compositions $r \in R_i$ but have higher frequencies, as well as all configurations that employ lines and frequencies as in $X_{\text{conflict}}$ but use different vehicles, which suffer from analogous conflicts.

4.2.2. Variants

Although (3) is generally tractable, for large instances this model is a large-scale MIP that requires substantial computational effort to be solved, even when using state-of-the-art commercial optimization solvers. Moreover, this model may need to be solved many times when iterating between LPP and TTP. For these reasons, we propose two heuristic simplifications to reduce computation time.

The first LPP simplification is called LPP-prune and aims at shrinking the MIP formulation by removing variables that are unlikely to be part of a solution. More specifically, recall that (3) needs flow variables $p_{e}^{od}$ for each $SG$ edge and demand relation, but $|S \times OD|$ can be a very large number. However, when routing a given $od \in OD$, typically only a small subset of $SG$ edges $s \in S$ will be part of a solution, i.e., where variables $p_{e}^{od}$ are strictly positive. Since the LPP minimizes passengers' travel time, these edges will belong to paths with short travel time, e.g., "closer" to the shortest path from $o$ to $d$. To exploit this intuition, we limit the maximal detour of each $od \in OD$ to a fixed percentage of the shortest path in the $SG$ (e.g., no more than 35% longer) and remove from the model (equivalently, fix to zero) all flow variables that do not fulfill this limitation. Moreover, we restrict the stations unsuitable for passenger transfers by removing their change edges in the $SG$. We will show in Section 5 that this simplification drastically reduces solving time while preserving solution quality.
In the second LPP variant, named LPP-simple, we set the duration \( t_i \) of all wait edge \( s \in S_{\text{worst}} \) based on the maximal frequency of the corresponding line \( l \), i.e., \( t_i := T/(2 \cdot \max(f : f \in F_l)) \). This modification helps reducing computation time as it decouples travel times \( t_i \) from frequencies. Moreover, frequency-dependent waiting times favor the selection of line pool candidates with high frequency but also lead to more infrastructure occupation and potentially more conflicts, as discussed in Yan and Goverde (2019). Thus, using LPP-simple instead of LPP yields line plans that suffer from fewer conflicts in the TTP. Note that the duration of the wait edges is set to its lowest possible value for each given line \( l \in L \). Moreover, each solution to LPP-simple is feasible to LPP and vice versa. This implies that solving the LPP-simple provides a feasible line plan and a lower bound to the optimal LPP objective. Therefore, we can solve LPP-simple instead of LPP until the resulting TTP is feasible, and as soon as this is the case, switch to LPP and use the solution found by LPP-simple to warm start the model. This ensures that the overall approach still provides an optimal solution to the LPP. We will show in Section 5 the benefit of using this model variant in combination with LPP-prune.

4.3. Solving the periodic timetabling problem

In the traditional TTP, the assignment of trains to tracks is predetermined or tackled afterwards as a separate routing and platforming problem (Caimi et al., 2017). Instead, we consider the routing of train services during the TTP, which produces a feasible timetable while accounting for the available resources. Note that this adds significant value to our approach, especially when dealing with large and busy networks, as we can precisely exploit the available infrastructure capacity. In the following, we initially formulate the TTP as an MIP (Section 4.3.1) before encoding it as an SAT (Section 4.3.2).

4.3.1. MIP formulation

Underlying our TTP formulation is an event activity network \( EAN = (E, A) \), where nodes \( e \in E \) represent events (e.g., a train arriving at a station or departing from it) and edges \( a \in A \) represent four types of activities (Lieberherr and Möhring, 2004): (i) trip activities model trains running between stations, (ii) dwell activities take place within stations, (iii) headway activities separate trains using the same infrastructure for safety and technical reasons, and (iv) frequency activities ensure an even spacing of consecutive services when a line is operated with frequency larger than one.

To construct such an EAN, we take the optimal line plan \( X_{\text{plan}} \) resulting from the LPP, and consider the lines \( L' \subseteq L \) that are part of this plan. Recall from sections 4.1–4.2 that the route section (RS) describes stations where a line stops, and the route itinerary (RI) is a DAG that encodes precisely the railway infrastructure and routing options available. At a high level, the EAN is obtained by considering information available from both the RS and RI, which models all lines that appear in \( L' \). We also shrink these graphs by introducing a clustering approach described next.

Modeling lines in \( L' \) which are considered to run in \( X_{\text{plan}} \) with frequency \( f > 1 \) requires the use of frequency activities (i.e., edges in the EAN). However, if we were to add such an activity for every combination of two arrivals at a station with \( n \) tracks, we would need to add \( O(n^2) \) activities even though eventually we require exactly one. Thus, the formulation would suffer from superfluous frequency edges. We address this issue by clustering all arrival events and departure events for each station and RI, as shown in Fig. 7 for one line. Note that this clustering technique does not restrict the solution space as exactly one train uses one RI and the RI is a DAG. Thus, the flow at any valid cut on the RI equals 1, and if we require all tracks of a station to form a valid cut, then exactly one event in the cluster is activated when the itinerary is assigned. In conclusion, with clustering, only one arrival and departure event is needed per station and RI, and only one frequency activity per pair of services.

The aim of the TTP is to schedule all events \( e \in E \) with a timestamp \( t_e \in [0, T) \) such that all activities \( a \in A \) are assigned with a duration \( \tau_a \geq 0 \) that is valid within a given lower bound \( l_a \) and upper bound \( u_a \), i.e., \( l_a \leq t_e \leq u_a \). The duration \( t_e \) of an activity \( a \in A \) corresponds to the time difference between its origin event \( e_{\text{or}} \in E \) and destination event \( e_{\text{de}} \in E \). In this sense, we keep the association between activities and edges; and between events and nodes, in the EAN. Given the periodicity of our TTP, we allow for \( \kappa_a \in \mathbb{Z} \) to adjust \( t_e \) by the period \( T \) if \( e_{\text{or}} \) takes places after \( e_{\text{de}} \), that is: \( t_e = t_{e_{\text{de}}} - t_{e_{\text{or}}} + \kappa_a T \).

In the EAN, we combine information about both Route Section RS characteristics, and Route Itinerary RI characteristics. By their design, the former is a single path for a service from its origin to destination, while the latter is a graph with multiple paths. In the solution process, we need to identify which specific infrastructure will be used by the chosen itinerary. We do this by selecting specific activities and events — and associated edges and nodes. Due to the routing component of our model, we want to be able to select an activity \( a \) only when it is associated to a chosen itinerary. To this end, we introduce the concept of selectable activity (and edge), associated to the RI, which means that its duration must be valid when it is part of a chosen itinerary. Note that this is more general than the formulation by Wüst et al. (2019) that allows selecting headway constraints if the corresponding track is chosen. We model a selectable activity \( a \in A \) with

\[
l_a - l_a \cdot y_a \leq t_a \leq u_a + (T - u_a) \cdot y_a,
\]

where \( y_a \in \{0, 1\} \) is a binary variable that can be activated \( (y_a = 0) \) or deactivated \( (y_a = 1) \). If \( y_a = 0 \), then the activity bounds \( l_a \leq t_a \leq u_a \) must hold. Instead, \( y_a = 1 \) implies \( 0 \leq t_a \leq T \), meaning that the activity is not restricted and both affected events can take place at any time.

We also define selectable events (and nodes) \( e \), which are associated to the RI. Similarly to above, a binary variable \( z_e \) determines whether those are selected or not, i.e., whether the chosen itinerary passes through them or not. We define the set of all selectable events as \( Z \subseteq E \).
We model the routing of a train over a line \( l \) as a binary flow problem in an \( RI \) network. When an edge is traversed, the corresponding activity must be selected and the events before and after take place. Conversely, when a node is not selected, no activity can be performed immediately after or immediately before the event (i.e., the edges starting from or ending to this node).

Henceforth, \( Z \subset E \) and \( Y \subset A \) define the selected nodes and edges, which can be associated to respectively selected events and activities. Those nodes and edges are associated to the chosen itinerary, as modeled at \( RI \) level in the \( EAN \). As an example of the selected nodes and edges for a single line, consider the portion of the graph made by the black arrows and all nodes in the bottom panel of Fig. 7. We indicate the set of the source nodes (i.e., without predecessors) of line \( l \in L' \) with \( Z^{source}_l \subseteq Z \). Finally, \( A_S := A \setminus Y \subset A \) contains only those activities which do not need to be selected, i.e., corresponding to the route sections of lines \( l \in L' \) (see, e.g., the blue arrows in the bottom panel of Fig. 7). The TTP with routing is formulated as an MIP in (4):

\[
\begin{align*}
\min & \quad \sum_{a \in A_S} p_a \cdot t_a \\
\text{subject to:} & \quad t_a \leq u_a + y_a (T - u_a - dt), \quad \forall a \in A, \quad (4a) \\
& \quad t_a \geq l_a - y_a \cdot l_e, \quad \forall a \in A, \quad (4b) \\
& \quad z_{a^+} + z_{a^-} + 2y_a \leq 2, \quad \forall a \in Y, \quad (4c) \\
& \quad \sum_{a \in \delta^{+(e)}} y_a = \sum_{a \in \delta^{-(e)}} y_a = z_e, \quad \forall e \in Z, \quad (4d) \\
& \quad \sum_{z_e \in Z^{source}} z_e = 1, \quad \forall l \in L', \quad (4e) \\
& \quad t_e = (t_{e^-} - t_{e^+}) dt + k_e T, \quad \forall a \in A, \quad (4f) \\
\text{variables:} & \quad t_a \in \{0, 1, ..., (T - dt)/dt\}, \quad \forall e \in E, \quad (4g) \\
& \quad t_a \geq 0, \quad \forall a \in A, \quad (4h) \\
& \quad z_e, y_a \in \{0, 1\}, \quad \forall e \in Z, \forall a \in Y \quad (4i) \\
& \quad k_e \in \mathbb{Z}, \quad \forall a \in A \quad (4j)
\end{align*}
\]

The objective (4a) minimizes the weighted duration of the activities in \( A_S \), which is consistent with the objective (3a) of the LPP. Constraints (4b)–(4c) ensure that lower and upper bounds of selected activities are respected, while constraints (4d) link an activity \( a \) with the variable associated to its corresponding nodes of origin \( z_{a^+} \) and destination \( z_{a^-} \). Constraints (4e) enforce flow conservation in the network, requiring that if \( e \) is active \( z_e = 1 \) then one of the outgoing and incoming edges must be active too. The presence of a flow is ensured by (4f), which demands that exactly one of the binary variables of the source nodes \( Z^{source}_l \) is chosen to be one. Since all \( z_e \) and \( y_a \) are binary variables, exactly one path per line in the \( RI \) is mandatory. Besides, as the \( RI \) is a DAG, no sub-tours or truncated paths may occur.

Note that model (4) uses integer timestamps \( t_a \) based on a discretization parameter \( dt \), which is common in literature on periodic timetabling (Liebchen and Möhring, 2004). This choice is also necessary to encode the TTP as an SAT. Due to periodicity, the origin event \( e_{a^+} \) of an activity \( a \) may take place after the target event \( e_{a^-} \). In such case, the difference is adjusted by \( k_e T \) in (4g), with \( k_e \in \mathbb{Z} \). To strengthen the formulation, we bound \( k_e \) by using the cuts proposed by Odijk (1996).
4.3.2. SAT formulation

We proceed to encode the TTP as a Boolean satisfiability problem. During the encoding, we convert the TTP into an equivalent formulation in conjunctive normal form, where clauses represent Boolean expressions involving literals that can be true or false, and denoted \( q \). A conjunctive normal form is satisfiable if at least one valid assignment for all literals exists. For further information on SAT, we refer to Schöning (2013). The encoding is motivated by two reasons. First, in periodic timetabling, SAT formulations are known to outperform MIP formulations when determining feasibility (Kümmling et al., 2015). Second, as we expect to face instances where no feasible timetable exists, efficiently determining an \( X_{\text{conflict}} \) is critical. Indeed some SAT solvers provide the unsatisfiable cores and we can exploit this information to identify the services corresponding to the literals in such cores. Note that any solution found by means of SAT can be turned into an MIP solution, and vice versa, as demonstrated by Borndörfer et al. (2020) for the case of TTP without routing. In the following, we consider the SAT encoding by Großmann (2011) and enhance it to assign train itineraries.

We initially order-encode all events \( e \in \mathcal{E} \) such that timestamps \( t_e \) are assigned to literals \( q \). (5).

\[
\text{encode-event}(e) := (\neg q_{e,-dt} \land q_{e,T-dt} ) \bigwedge_{i \in \{0,dt,\ldots,T-dt\}} (\neg q_{i,j-dt} \lor q_{i,j}).
\]

(5)

In our formulation, a selectable activity \( a \in \mathcal{A} \) has to be valid only when the literals \( q_{zi} \) and \( q_{zj} \), associated respectively to the starting node \( i = e_i \in \mathcal{Z} \) and ending node \( j = e_j \in \mathcal{Z} \) are activated. Thus, we enhance the Boolean expression encode-rectangle from Großmann (2011) and define in (6) the new expression encode-selectable-rectangle.

\[
\text{encode-selectable-rectangle}([i_1,i_2] \times [j_1,j_2]) := \neg q_{i_2,j_2} \lor q_{i_1,j_1} \lor \neg q_{j_1,j_2} \lor q_{i_1,j_1} \lor \neg q_{i_2,j_1} \lor \neg q_{i_1,j_2}.
\]

(6)

Next, we encode the selectable nodes \( e \in \mathcal{E} \) of the RI graphs directly to SAT literals by using the variables \( z_e \) associated to one node to a literal \( q_e \), and encode the selectable edges \( a \in \mathcal{Y} \) via direct mapping of the associated variable \( y_a \) to a literal \( q_a \), as well. By applying (7) to all nodes \( e \in \mathcal{E} \), we enforce flow conservation in the RI graphs for all lines \( l \in \mathcal{L}' \). Specifically, to ensure that exactly one path per RI is activated, we add an encoded source node \( q_{\text{source}} \) and sink node \( q_{\text{sink}} \) to the RI, and request that both \( q_{\text{source}} \) and \( q_{\text{sink}} \) are true in the assumption literals. Expression (7) enforces a literal \( q_e \) to be true if any of the incoming edges \( q_e \in \delta^-(z) \) is activated. Furthermore, if \( q_e \) is active, it will also require one of the outgoing \( q_e \in \delta^+(z) \) to be true. To make sure that in any case at most one incoming and one outgoing edge is true, we use encode-at-most-one (8) as a helper, which allows at most one literal \( q \) of a set \( Q \) to be true.

\[
\text{encode-RI}(z) := \left( \neg q_z \lor \bigvee_{y \in \delta^+(z)} q_y \right) \land \left( \neg q_z \lor \bigvee_{y \in \delta^-(z)} q_y \right) \land \\
\text{encode-at-most-one}(\delta^+(z)) \land \text{encode-at-most-one}(\delta^-(z)).
\]

(7)

\[
\text{encode-at-most-one}(Q) := \bigwedge_{e, q_e \in Q, i < j} (\neg q_i \lor \neg q_j).
\]

(8)

Although SAT is superior to MIP when finding initial solutions for the TTP, large-scale instances may still be challenging to solve (Kümmling et al., 2015). Consequently, we propose an iterative fix-add-and-relax approach, which is based on the idea of Herrigel et al. (2018) but distinguishes itself by using the information of unsatisfiable cores provided by an incremental SAT solver. Specifically, using the unsatisfiable core, we locally relax the tightened events or extract an \( X_{\text{conflict}} \) if no more relaxations are possible. We eventually employ two decomposition techniques to supplement the fix-add-and-relax approach: the first decomposes the instance by services in two levels of depth; the second uses a partition of the \( E\!\!A\!N \) by route constraint to search for conflicts locally.

5. Numerical study

In this section, we test our approach in an extensive computational study. We present the considered real railway instances in Section 5.1. Then, we investigate the LPP and TTP modules independently in Section 5.2 and Section 5.3, respectively. Finally, we assess the performance of the full iterative method in Section 5.4.

5.1. Real-life test case and configurations

We consider the portion of the Swiss railway network operated by Rhätische Bahn (RhB), which is illustrated in Fig. 8 and consists of 384 km of tracks and 102 stations (RhB, 2021). The infrastructure is defined at a mesoscopic level of details, and because more than 80% of the sections are single-track sections, it is particularly challenging to find conflict-free timetables.

Since RhB operates the network with an hourly periodicity, we set the time interval \( T \) equal to one hour. Moreover, RhB provided us with the passenger demands in the form of trips in the network on an average workday. As the demands in our model should be consistent with the period \( T \), we scale the daily relations by 0.1 to approximate the demand during a typical hour. In total, we have 1763 origin–demand pairs, which are roughly 17.5% of all possible pairs among the 102 stations in the network. To investigate the performance of the proposed method at varying instance size, we define three demand scenarios. A small demand scenario \( \mathcal{OD}_S \) contains only the relations between the stations of Arosa and Chur, such that \(|\mathcal{OD}_S| = 2\). In a medium scenario \( \mathcal{OD}_M \), we remove 50% of the relations, namely all those with less demand than the median number of passengers across relations, which
Fig. 8. Network of Rhätische Bahn with an excerpt of the infrastructure.

F. Fuchs et al.

Transportation Research Part C 142 (2022) 103805

The line pool we consider is based on the currently operated lines and possible modifications thereof. It includes mixed services, with four freight services and the car shuttle through the Vereina tunnel. There are currently 10 operated lines, while the line set \( \mathcal{L} \) we consider consists of 47 lines. The non-public services have a single frequency and are mandatory to operate, while the others may be operated at higher frequencies. We use a constant speed per train category to approximate the traversal times: 19 m/s for cargo services, 20 m/s for regional trains, and 21 m/s for local and express trains, as this choice leads to trip times that match well with the actual timetable data (SBB, 2021). There are 8 distinct types of rolling stock units \( G \), each available in different number, for a total of 465 units. According to the current line plan, we consider 11 compositions of this rolling stock, providing a capacity \( \text{cap} \) between 120 and 300 for passenger services. Some configurations are instead specific to freight. Under these specifications, the size of the line pool is \(|\mathcal{X}_{\text{pool}}| = 472\).

In preliminary experiments, we observed that without any restriction on the total frequencies \( f_{i, \text{max}} \) on the various track sections, the LPP generates line plans with a considerable number of conflicts. Therefore, we define two restrictions on \( f_{i, \text{max}} \) applied to all sections \( i \in \mathcal{I} \). The first restriction \( \text{restrict-6-14} \) allows for 6 and 14 trains per single and double track section, respectively, while the second \( \text{restrict-8-16} \) allows for 8 and 16 trains. Concerning the LPP-prune, we restrict the maximal detour for passengers to be at most 35% longer than the shortest connection. Besides, in this model we restrict the stations suitable for changes according to the current line plan (RhB, 2021).

In the TTP, we use a time discretization \((dt)\) of 20 s. Note that the most common time discretization used in the literature is 60 s (Liebchen and Möhring, 2004) but we opt for a higher granularity to improve accuracy and reduce the impact of rounding errors (at the obvious expense of a larger model). The minimal headway \( h \) is set to 120 s for the entire network based on current practice (RhB, 2020). The feasibility of the TTP is significantly affected by the activity bounds \( l_a \) and \( u_a \) and how close they are to each other. Consequently, we define three scenarios that incorporate different levels of slack in the bounds of trip and dwell activities, which influence the flexibility in the TTP to adjust the duration of such activities. These scenarios are called no-slack, some-slack, and max-slack, and are specified in Table 3.

The framework is implemented using Python 3.8 and relies on the Python-igraph library (Csardi and Nepusz, 2006). We solve all MIP models with the commercial solver Gurobi 9.1.0 via the Python interface. For the incremental SAT solving, we use Glucose 4.1 (Audemard and Simon, 2018), which we access via the pysat module provided by Ignatiev et al. (2018). To solve the partitions in the two SAT decomposition approaches mentioned in Section 4.3.2, we use CryptoMiniSat 5.8.0 (Soos et al., 2009). Glucose 4.1 is single-threaded while the decomposition approaches are each given five threads.
5.2. Assessing the line planning module

We examine the LPP module by solving model (3) with a time limit of one hour and a target optimality gap of 0.5%. We also investigate the effect of the heuristic variants LPP-prune and LPP-simple introduced in Section 4.2.2, as well as a third variant named LPP-prune-simple that simultaneously accounts for both features of pruning and constant access times. The results for the three demand scenarios OD_s, OD_M, and OD_L are reported in Table 4, without using frequency limitations \( f_{\text{max}} \).

The OD_s instance does not impose any challenge to any of the four models, that is, all models lead to the same objective function value with a computation time of less than half a second. The situation changes drastically with the larger demand scenarios OD_M and OD_L. Specifically, if we compare the results by LPP with those by LPP-prune, we notice that the two models provide almost identical objective values, with a difference that is smaller than 0.2%. However, LPP-prune solves both scenarios in about 20–30 s whereas LPP reaches the one-hour time limit without meeting the required 0.5% optimality gap threshold. This suggests that LPP-prune (with an allowed 35% detour) can be used in lieu of LPP to speed up execution substantially without decreasing quality.

Finally, the line planning models that employ constant waiting/access times (i.e., LPP-simple and LPP-prune-simple) are significantly faster to solve than the respective models that instead use frequency-dependent times (i.e., LPP and LPP-prune). In particular, LPP-prune-simple solves OD_s and OD_M in 2–5 s. The objective function decreases by about 9% in the simplified models as they provide lower bounds as discussed in Section 4.2.2. Overall, we can say that our intentions to shrink and simplify the models are proven effective; hence, our iterative approach will employ LPP-prune-simple to find feasible conflict-free solutions before switching to LPP-prune.

5.3. Assessing the timetabling module

We assess the TTP module on nine instances that combine demand scenarios, slack parameters, and total frequency restrictions. For each instance, the first line plan provided by the LPP module is used as input to the TTP. We compare three different approaches for timetabling: (i) TTP-SAT solves the SAT formulation (5)–(8) by concurrently applying the fix-add-and-relax method as well as two decomposition techniques (see Section 4.3.2), (ii) TTP-MIP solves the MIP formulation (4) directly using a solver, and (iii) TTP-fixed simplifies the timetabling problem by neglecting the routing of services through their RI. More specifically, we fix all itineraries a priori based on a leftmost-path rule that guides a train via the leftmost nodes of the RI in the direction of travel. The results of these experiments are summarized in Table 5, which reports computation time and a feasibility indicator.

Since TTP-SAT determines whether an instance admits a feasible timetable in all cases, it provides the ground truth. Moreover, across the nine instances considered, this approach runs in roughly 30 s on average, and always in less than 2 min, also for the larger instances. In contrast, TTP-MIP exceeds the one-hour time limit in three out of nine cases, thus not being able to consistently establish feasibility. Based on these results, TTP-SAT appears superior over TTP-MIP to determine feasibility, which is aligned with earlier findings in the literature for a timetabling problem that does not include routing (Börndörfer et al., 2020). Thus, we employ TTP-SAT in our iterative algorithm.

Finally, TTP-fixed provides another interesting insight as it concludes infeasibility for all nine timetable instances, although five are actually feasible when allowing for routing. In fact, this approach uses predefined routes and can only establish feasibility for the chosen itineraries. Therefore, an instance labeled as infeasible by TTP-fixed would require further investigation. This shows the benefit of accounting for routing in the TTP, whilst relying on fixed itineraries may be too restrictive.
These sizes are much smaller than the number of lines in the banned Table 7. However, whilst banning X when banning \( \Sigma \) approach that bans conflicts converges in about 20 and 60 min, respectively, for feasible timetable is found within 10 h of execution when applying the existing strategy that bans line plans. In contrast, our novel demand scenarios, and display the results of the comparison in Table 7 with a focus on the convergence of the approach.

conflicts (complicating features such as a high number of single-track sections.
timetables. Moreover, they show that our iterative approach is able to efficiently tackle real railway instances with potentially higher number of services per single and double-track section is allowed here, at most seven conflicts are detected before a feasible max-slack for higher flexibility in scheduling activities (activity bounds is allowed (i.e., timeout).

\( \max\text{-slack} \) gives the best objective function per \( \OD_h \) scenario. Furthermore, despite a higher number of services per single and double-track section is allowed here, at most seven conflicts are detected before a feasible timetable is found and in less than an hour. These results underscore that incorporating some flexibility is critical to obtain feasible some-slack. In contrast, when some slack is allowed (some-slack), the iterative approach eventually yields a feasible timetable but requires up to 40 iterations and 5 h of execution. The bottom portion of the table shows that allowing varying conditions related to the demand set, slack parameters, and maximum line frequencies. We report the detailed results of our experiments in Table 6.

As shown in the top portion of this table, only the small demand scenario leads to a feasible timetable when no slack on the activity bounds is allowed (no-slack). In contrast, when some slack is allowed (some-slack), the iterative approach eventually yields a feasible timetable but requires up to 40 iterations and 5 h of execution. The bottom portion of the table shows that allowing for higher flexibility in scheduling activities (max-slack) gives the best objective function per \( \OD_h \) scenario. Furthermore, despite a higher number of services per single and double-track section is allowed here, at most seven conflicts are detected before a feasible timetable is found and in less than an hour. These results underscore that incorporating some flexibility is critical to obtain feasible timetables. Moreover, they show that our iterative approach is able to efficiently tackle real railway instances with potentially complicating features such as a high number of single-track sections.

5.4. Assessing the overall iterative approach

The last series of experiments involves the overall iterative approach (see Section 3.1 for an overview) for nine instances under varying conditions related to the demand set, slack parameters, and maximum line frequencies. We report the detailed results of our experiments in Table 6.

To complement the theoretical assessment provided in Section 3.4, we compare numerically the two iterative strategies that ban conflicts (\( \lambda_{\text{conflict}} \)) and ban infeasible line plans (\( \lambda_{\text{plan}} \)). To this end, we tested the max-slack & restrict-8-16 setting with all the three demand scenarios, and display the results of the comparison in Table 7 with a focus on the convergence of the approach.

While convergence is similar for the small demand scenario \( \OD_h \), the effect of banning \( \lambda_{\text{conflict}} \) is remarkable as it reduces the number of iterations by 25 and 12 times under the more realistic scenarios \( \OD_M \) and \( \OD_L \), respectively. In these scenarios, no feasible timetable is found within 10 h of execution when applying the existing strategy that bans line plans. In contrast, our novel approach that bans conflicts converges in less than 1 h for the small demand scenario and in less than one hour for the large demand scenario.

Fig. 9 shows the quality (i.e., value of the LPP solution) over the iterations for the two instances \( \OD_M \) and \( \OD_L \) analyzed in Table 7, when banning \( \lambda_{\text{conflict}} \) and \( \lambda_{\text{plan}} \). For all four cases, we can observe the moment when we switch the LPP model due to the first increase in the objective value. However, whilst banning \( \lambda_{\text{conflict}} \) converges towards a feasible line plan, banning \( \lambda_{\text{plan}} \) leads to an objective function value that remains steady over iterations and does not converge in our 10-hour limit, as was also shown in Table 7.

We further investigated the properties of the conflict sets and line plans found during the iterative procedure when banning \( \lambda_{\text{conflict}} \) or \( \lambda_{\text{plan}} \). To this end, we focus on instance \( \OD_M \) in Table 7 and start by analyzing the size of the banned conflicts and line plans. When looking at \( \lambda_{\text{conflict}} \), we observe that the size of this set generally varies between two to five conflicting lines over iterations. These sizes are much smaller than the number of lines in the banned \( \lambda_{\text{plan}} \) that mostly varies between 13 and 18. This
Table 7  
Banning line plans versus conflicts for the instance max-slack & restrict-8-16.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Iterations</th>
<th>$\sigma D_h$</th>
<th>$\sigma D_M$</th>
<th>$\sigma D_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banning $\mathcal{X}_{plan}$</td>
<td>6</td>
<td>&gt;179</td>
<td>&gt;83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running time [s]</td>
<td>56</td>
<td>&gt;36'000</td>
<td>&gt;36'000</td>
</tr>
<tr>
<td>Banning $\mathcal{X}_{conflict}$</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running time [s]</td>
<td>41</td>
<td>1171</td>
<td>3539</td>
</tr>
</tbody>
</table>

Fig. 9. Evolution of the objective function value when banning conflicts and line plans.

Further underscores that $\mathcal{X}_{conflict}$ contains a small subset of $\mathcal{X}_{plan}$, which is highly beneficial for convergence consistently with the theoretical results of Section 3.4.

When looking at the first conflict set $\mathcal{X}_{conflict}$ generated with our approach, we observe that this conflict appears in four line plans $\mathcal{X}_{plan}$ that are subsequently found by the benchmark approach that bans line plans. As banning this $\mathcal{X}_{conflict}$ would have banned all four $\mathcal{X}_{plan}$, we can say that we have saved at least three iterations concerning this specific conflict. More in general, for the 11 different $\mathcal{X}_{conflict}$ determined throughout the iterations, we can observe that each of them reoccurs between 1 and 165 times in the subsequent $\mathcal{X}_{plan}$, with an average of 31 recurrences (note that an $\mathcal{X}_{plan}$ can be affected by more than one $\mathcal{X}_{conflict}$).

We also noticed that some of the detected conflicts can be clustered geographically in narrow parts of the network, suggesting a direction of further enhancement. In particular, future research could provide a methodology to generalize a detected conflict and thereby effectively ban other similar configurations simultaneously. While such a generalization would possibly reduce even more the number of iterations, it is not straightforward to define due to the different routes available, dwell and trip times, and track assignments.

Eventually, when comparing pairs of $\mathcal{X}_{plan}$ identified during two subsequent iterations, we noticed that in about 60% of the 179 iterations, the two $\mathcal{X}_{plan}$ only differ by one line and frequency combination. Hence, observing the same $\mathcal{X}_{conflict}$ in more than one $\mathcal{X}_{plan}$ is to be expected. For the same reason, the structure of the corresponding timetables is often similar.

Since our case study considers a railway network, a graphical representation of the timetable yields limited insight (such representations are common for individual corridors). Nevertheless, to visually assess our results, we display the line plan for a representative instance $\sigma D_M$ with some-slack and restrict-6-14, and compare it with the line plan currently operated by RhB in Fig. 10. Interestingly, the two plans do not differ drastically from a graphical perspective. However, our solution leads to higher infrastructure utilization as generally more trains run. Furthermore, some lines are operated at a frequency of $f = 2$ in our solution (depicted as thick lines in the figure), whereas in the current plan all lines operate once per hour. This is possible as the optimized line plan is able to increase the frequency on specific, heavily demanded connections by carefully synchronizing operations and run more frequent services, despite the single track.

6. Conclusion

Motivated by the general need to achieve a more effective and integrated public transport planning, we considered in this paper the joint LPP and TTP in railways. We proposed an iterative approach based on an enhanced feedback loop between the two tasks. In particular, by extracting the smallest set of conflicting services from an infeasible TTP, and banning these conflicts in the LPP, our approach delivers a substantial theoretical and numerical advantage compared to existing methods that iteratively ban a line plan when the corresponding timetable is infeasible. A second key aspect of our work is that our model is aware of the available infrastructure thereby allowing for the routing and platforming of train services already at a planning stage. Here, we deviate from
(and generalize) the traditional TTP approach with given itineraries, and consider the task of assigning itineraries to trains while determining a periodic timetable.

We assessed the convergence of our approach theoretically and performed extensive numerical experiments involving real-life data of a railway network in Switzerland. Our results underline the crucial role of precise conflict detection in an iterative strategy for the joint LPP and TTP. Our method could efficiently generate line plans and timetables despite the many single-track sections, and converged in up to 25 times fewer iterations and 4% of the computation time when compared with a strategy that bans line plans. In addition, our approach proved to be robust towards different demand scenarios and parameter settings. We finally showed that allowing for flexibility in selecting train itineraries and trip times can significantly ease retrieving a feasible timetable.

The present paper deals with searching for a feasible timetable, given an optimal line plan. The published approaches for solving the timetabling problem indeed went over time from feasible solutions to the consideration of an optimization target (Liebchen and Möhring, 2004). A similar extension could be included in the proposed research, including figures of merit in the timetabling problem, possibly related to the travel times, utilization of infrastructure, usage of vehicles and crew resources, and other passenger indicators. Those adjustments can also take place at multiple levels and in an iterative fashion when the impact of the optimality of one layer on the other layers is clarified and understood (Goverde et al., 2016; Corman, 2020). Furthermore, Benders decomposition showed promising potential to solve similarly decomposed problems (Lamorgese et al., 2016; Leutwiler and Corman, 2022). Moreover, future research may involve enhancing the TTP module to consider both vehicle rotations and connections as done by Fuchs and Corman (2019); however, this opens up extra challenges in the interaction between the two modules. Another avenue would be the inclusion of robustness in the TTP the as proposed by Caimi et al. (2011), which would add further practical value to the resulting timetable. Finally, as some subsets of lines share some of their sections regularly, one could try to leverage the information of an identified conflict to catch other conflicts affecting related lines, similarly to what was proposed by van Lieshout et al. (2020).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank RhB for providing us with data for the numerical experiments and feedback on our study. We are also grateful to Florin Leutwiler for valuable inputs and comments on the model, and to two anonymous referees for feedback that helped us improve the manuscript. This work is partly supported by the Swiss National Science Foundation under Project 1481210/DADA, and is based on an extended abstract that has been accepted for presentation at CASPT 2021.

Appendix A. List of symbols

The Table 8 provides an overview of the nomenclature (e.g., parameters and decision variables) used in the different models throughout the paper.
Appendix B. Algorithm for data structure generation

In this appendix, we provide the pseudocode of the algorithm used to derive all data structures that are needed as input to the timetabling and line planning problems (RC, RS, RI, and EAN). The core elements of this pseudocode are the three functions mentioned below:

- **FindAllStationNodes.** This function finds, among the set of microscopic infrastructure elements, all nodes which happen to be associated with a given station.
- **FindAllRoutes.** This function computes all routes in the microscopic infrastructure specification between any two pairs of nodes.
- **Generate.** This function creates a node data structure, with given name, which can be further connected to predecessor and successor nodes.

### Parameters and elements of the data structures

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>Time period considered</td>
</tr>
<tr>
<td>$od ∈ Od$</td>
<td>One origin-destination pair, from the set of all origin-destination pairs</td>
</tr>
<tr>
<td>$y^{od}$</td>
<td>Amount of passenger demand for $od$</td>
</tr>
<tr>
<td>$(l, f, r) ∈ X_{pool}$</td>
<td>An item of the line pool, which is the set of all candidate items to build a line plan</td>
</tr>
<tr>
<td>$l ∈ L$</td>
<td>A line for a candidate item in the line pool, in the set of all possible lines</td>
</tr>
<tr>
<td>$f ∈ F_l$</td>
<td>A possible line frequency, in the set of all possible line frequencies</td>
</tr>
<tr>
<td>$r ∈ R_l$</td>
<td>A rolling stock configuration, in the set of all possible rolling stock configurations</td>
</tr>
<tr>
<td>$mxe_{g, r}$</td>
<td>Number of units of type $g$ used in configuration $r$</td>
</tr>
<tr>
<td>$g ∈ G$</td>
<td>A rolling stock unit type, in the set of all rolling stock unit types</td>
</tr>
<tr>
<td>$s_{max}$</td>
<td>Maximum amount of units of type $g$</td>
</tr>
<tr>
<td>$cap_p$</td>
<td>Passenger capacity for a rolling stock configuration $r$</td>
</tr>
<tr>
<td>$X_{conflict}$</td>
<td>Subset of line pool elements which jointly generate a conflict</td>
</tr>
<tr>
<td>$C$</td>
<td>Set of conflicts</td>
</tr>
<tr>
<td>$RC$</td>
<td>Route Constraint: sequence of visited stations</td>
</tr>
<tr>
<td>$RS$</td>
<td>Route Section: sequence of visited stations with a planned stop</td>
</tr>
<tr>
<td>$RI$</td>
<td>Route Itinerary: graph representing the available tracks</td>
</tr>
<tr>
<td>$SG = (V, S)$</td>
<td>Section Graph</td>
</tr>
<tr>
<td>$v ∈ V$</td>
<td>A node, in the set of all nodes in the $SG$</td>
</tr>
<tr>
<td>$s, d$</td>
<td>Node for origin ($s$) and destination ($d$) in the $SG$</td>
</tr>
<tr>
<td>$s ∈ S$</td>
<td>An edge, in the set of all edges in the $SG$</td>
</tr>
<tr>
<td>$S_{wait}$</td>
<td>Wait edges in the $SG$</td>
</tr>
<tr>
<td>$k_{slack}$</td>
<td>Multiplier of minimum running time due to time reserves</td>
</tr>
<tr>
<td>$t_s$</td>
<td>Minimum duration for edge $s$</td>
</tr>
<tr>
<td>$t_{speed}$</td>
<td>Minimum running time for edge $s$</td>
</tr>
<tr>
<td>$i ∈ I$</td>
<td>A section of railway track, in the set of all sections of railway tracks (=infrastructure)</td>
</tr>
<tr>
<td>$X_l$</td>
<td>Candidate items in the line pool, which use section of railway track $i$</td>
</tr>
<tr>
<td>$f_{max}$</td>
<td>Upper bound on total frequencies for section of railway track $i$</td>
</tr>
<tr>
<td>$EAN = (E, A)$</td>
<td>Event Activity Network</td>
</tr>
<tr>
<td>$e ∈ E$</td>
<td>An event, in the set of all events in the $EAN$</td>
</tr>
<tr>
<td>$a ∈ A$</td>
<td>An activity, in the set of all activities in the $EAN$</td>
</tr>
<tr>
<td>$l_a, u_a$</td>
<td>Lower and upper bound on the duration of activity $a$</td>
</tr>
<tr>
<td>$di$</td>
<td>Discretization parameter for $t_s$</td>
</tr>
<tr>
<td>$\delta^+(\cdot)$</td>
<td>Set of outgoing and incoming arcs to a node</td>
</tr>
<tr>
<td>$\delta^-(\cdot)$</td>
<td>Event (node) before (=origin of) and after (=destination of) activity (edge) $a$</td>
</tr>
</tbody>
</table>

### Decision variables in the LPP and TTP models

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{r,s}$</td>
<td>Binary variable to select an element of the line pool</td>
</tr>
<tr>
<td>$X_{plan}$</td>
<td>Line plan, which is the set of selected elements of the line pool</td>
</tr>
<tr>
<td>$p_{od}^s$</td>
<td>Continuous variable to choose the amount of demand for $od$ using edge $s$</td>
</tr>
<tr>
<td>$L'$</td>
<td>Set of lines $l$ which appear in an item in $X_{plan}$</td>
</tr>
<tr>
<td>$t_e$</td>
<td>Discretized variable to set the timestamp for event $e$</td>
</tr>
<tr>
<td>$t_a$</td>
<td>Continuous variable to set the duration of activity $a$</td>
</tr>
<tr>
<td>$y_a$</td>
<td>Binary variable equal to 1 when activity $a ∈ Y$ is selected and takes place</td>
</tr>
<tr>
<td>$k_a$</td>
<td>Integer variable denoting the time period multiplier</td>
</tr>
<tr>
<td>$z_a$</td>
<td>Binary variable equal to 1 when event $e ∈ L$ is selected and takes place</td>
</tr>
<tr>
<td>$Z$</td>
<td>Selectable activities in $A$ from $RI$ of lines in $L'$</td>
</tr>
<tr>
<td>$A_X$</td>
<td>Non-selectable activities in $A$ from $RS$ of lines in $L'$</td>
</tr>
<tr>
<td>$Z_{source}$</td>
<td>Source node of lines in $L'$ in $RI$</td>
</tr>
<tr>
<td>$q ∈ Q$</td>
<td>A generic literal, in the set of all literals in SAT</td>
</tr>
<tr>
<td>$q_{l,i}$</td>
<td>Literal in SAT identifying if event $e$ takes place at time $i$</td>
</tr>
<tr>
<td>$q_{source}, q_{sink}$</td>
<td>Additional SAT literals, modeling one source and sink node for each path in $RI$</td>
</tr>
</tbody>
</table>
Algorithm 1: Generation of relevant data structures

\begin{verbatim}
Input : Set of lines \( l \in L \), each being a list of (stopping or not) stations visited;
        The infrastructure, described at a microscopic level

Output: A set of nodes and edges in the \( RC \);
         A set of activities and events in the \( RS \);
         A set of activities and events in the \( RI \)

Initialization: \( V_{RC} = \emptyset ; \; S_{RC} = \emptyset ; \; A_{RS} = \emptyset ; \; \mathcal{E}_{RS} = \emptyset ; \; A_{RI} = \emptyset ; \; \mathcal{E}_{RI} = \emptyset \)

foreach line \( i \in \mathcal{L} \)
    foreach two consecutive (stopping or not) stations \( i, j \in \mathcal{L} \)
        \( SI = \text{FindAllStationNodes}(i, \text{infrastructure}) \)
        \( SJ = \text{FindAllStationNodes}(j, \text{infrastructure}) \)
        \( P = \text{FindAllRoutes}(SI, SJ, \text{infrastructure}) \)
        \( V_{RC} := V_{RC} \cup \{i, j\} \)
        \( S_{RC} := S_{RC} \cup \{i \rightarrow j\} \)
        if (\( l \) stops at \( i \))
            Generate \( arr_r, dep_r \)
            \( \mathcal{E}_{RS} := \mathcal{E}_{RS} \cup \{arr_r, dep_r\} \)
            \( A_{RS} := A_{RS} \cup \{arr_r \rightarrow dep_r\} \)
            \( dep_p = dep_r \)
        end if
    end for (each \( P \))
end for (each \( i,j \))

foreach track-element \( t \in p \)
    if (\( t \) stops at \( i \))
        Generate \( arr_r, dep_r \)
        \( \mathcal{E}_{RI} := \mathcal{E}_{RI} \cup \{arr_r, dep_r\} \)
        \( A_{RI} := A_{RI} \cup \{t \rightarrow arr_r\} \cup \{arr_r \rightarrow dep_r\} \)
        \( r = dep_r \)
    else
        \( \mathcal{E}_{RI} := \mathcal{E}_{RI} \cup \{t\} \)
        \( A_{RI} := A_{RI} \cup \{t \rightarrow t\} \)
        \( r = t \)
    end if
end for (each \( t \))
end for (each \( p \))
end for (each \( i,j \))
end for (each \( l \))

\( RC := (V_{RC}, S_{RC}) \)
\( RS := (\mathcal{E}_{RS}, A_{RS}) \)
\( RI := (\mathcal{E}_{RI}, A_{RI}) \)
\( EAN := RS \cup RI \)
\end{verbatim}

References


F. Fuchs et al.

\[ E_A N \; = \; (R_I, S_{I,J}, P) \]
\[ R_C \; = \; (V_{RC}, (S_{RC}, E_{RS}, A_{RS})) \]
\[ R_I \; = \; (E_{RI}, A_{RI}) \]
\[ E_A N \; = \; RS \cup RI \]

Transportation Research Part C 142 (2022) 103805