Towards Solving Very Large Scale Train Timetabling Problems by Lagrangian Relaxation

Frank Fischer, Christoph Helmberg
Chemnitz University of Technology

Jürgen Janßen, Boris Krostitz
Deutsche Bahn AG, Konzernentwicklung, GSU 1
Problem description

Classical Train Timetabling Problem (TTP).

**Goal:** generate a timetable for the *whole German railway network* of Deutsche Bahn.
Problem description

Classical Train Timetabling Problem (TTP).

**Goal:** generate a timetable for the *whole German railway network* of Deutsche Bahn.

**Given:**
- railway network (stations, tracks, track switches ...)
- passenger and freight trains with predefined route

**Restrictions:**
- running times, headway times, capacities
- base timetable for passenger trains

**Goal:**
- feasible timetable with few delays
The TTP is a well investigated problem:

- **periodic scheduling literature**
  - Serafini, Ukovich (1989)
  - Kroon, Dekker, Michiel, Vromans (2005)
  - Liebchen (2006)

- **non-periodic scheduling literature**
  - Schrijver, Steenbeck (1994)
  - Higgins, Kozan, Ferreira (1997)
  - Caprara, Fischetti, Toth (2002)
  - Cacchiani, Caprara, Toth (2006)
  - Caprara, Kroon, Monaci, Peeters, Toth (2006)
  - Borndörfer, Schlechte (2007)
Given:

- *infrastructure digraph* \( D = (V, A) \) where
  - \( V \) set of stations, track switches ...
  - \( A \) set of tracks, may be
    - *double tracks*
    - *single tracks*
  - *absolute node capacities*
  - *directional capacities*
Problem data

Example: absolute and directional capacities.

```
1 2
  3

2 3
  4
```

```
<table>
<thead>
<tr>
<th>absolute</th>
<th>left dir.</th>
<th>right. dir.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>absolute</th>
<th>left dir.</th>
<th>right. dir.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```
Trains

For each train $j \in T$:

- **train type** $m(j) \in M$

- **predefined route**: ordered sequence of nodes $U(j) = (u^j_1, \ldots, u^j_{n_j}), n_j \in \mathbb{N}$

Furthermore for each passenger train

- **stopping interval** $l^j_i = \left[ t^S_{i,j}, t^{E,j}_i \right] \subset \mathbb{Z} \cup \{ \pm \infty \}$
  “when the train has to wait”

- **minimal stopping time** $d^j_i \in \mathbb{Z}^+$.  
  “how long the train has to wait”

train must arrive before $t^{E,j}_i$ and must not leave before $t^{S,j}_i + d^j_i$.  

Stopping interval and minimal stopping time

Example: stopping interval = [1, 5], minimal stopping time = 2 minutes
The following examples are valid:
Running times

A train needs some time from one station to the next, its *running time*. This depends on the train type \((m \in M)\) and on whether the train *stops* or *passes* at the stations:

\[
t_a^R : M \times B_R \to \mathbb{Z}_+, a \in A, B_R = \{\text{pass, stop}\}
\]
Headway times

There must be a safety distance between two sequent trains on the same track, the *minimal headway times*.

\[
t_1 + t_a^H \leq t_2
\]

They depend on both train-types and stopping behaviours:

\[
t_a^H : M \times B_R \times M \times B_R \rightarrow \mathbb{Z}_+.
\]
Model

Classic model via \textit{time discretised networks} for the single train routes (e.g. Caprara et al.):
For each train $j \in T$ a graph $G^j = (V^j, A^j)$ where

- $V^j$ contains
  - an artificial \textit{start-node} $\sigma^j$,
  - an artificial \textit{end-node} $\tau^j$,
  - a \textit{wait-node} and a \textit{stop-node} node for each \textit{station}, \textit{time-step}
- $A^j$ contains
  - \textit{starting arcs} from $\sigma^j$ to the first station’s nodes,
  - \textit{ending arcs} from the last station’s nodes to $\tau^j$,
  - \textit{waiting arcs} between two successive wait-nodes of one station,
  - \textit{running arcs} connecting nodes of successive stations
  - \textit{infeasible arcs} from each intermediate station’s node to $\tau^j$. 
Train graphs: nodes

Station 2  Station 3

- t=1
- t=2
- t=3
- t=4
- t=5
- t=6
- t=7

- stop nodes train stops at the station
- run nodes train passes through the station

nodes
Train graphs: waiting arcs

waiting arcs

Station 2  Station 3
Train graphs: running arcs

**run-run**

**run-stop**

**start-run**

**start-stop**

Running arcs
Train graphs: infeasible arcs

infeasible arcs
Variables

Let $\mathcal{A} \coloneqq \bigcup_{j \in T} \mathcal{A}^j$ be the set of all arcs. Introduce binary variables for each arc:

$$x_a \in \{0, 1\}, \ a \in \mathcal{A},$$

with the interpretation for $a \in \mathcal{A}^j$:

$$x_a = 1 \iff \text{train } j \text{ uses arc } a.$$
Capacity constraints

Only a bounded number of trains may enter an infrastructure node \( v \in V \) at the same time \( t \) because of absolute capacities and directional capacities. Lead to constraints of the form

\[
\sum_{a \in \delta^-(v, t)} x_a \leq c_v, \quad \text{absolute capacities}
\]

and

\[
\sum_{a \in \delta^-(uv, t)} x_a \leq c_{uv}, \quad \text{directional capacities}
\]

where

\[
\delta^-(v, t) = \left\{ \left( (b', i', t')^j, (b, i, t)^j \right) \in \mathcal{A} : u_i^j = v \right\},
\]

\[
\delta^-(uv, t) = \left\{ \left( (b', i', t')^j, (b, i, t)^j \right) \in \mathcal{A} : u_{i-1}^j u_i^j = uv \right\}.
\]
Capacity constraints

Example: station 42 has capacity 1

\[
\sum_{a \in \{\text{red arcs}\}} x_a \leq 1.
\]
Headway constraints

Between two trains on the same physical track minimal headway times are required for safety reasons (e.g. Lukac).

Two arcs

\[((b_1, i_1, t_1)^j, (b_2, i_2, t_2)^j) \in A^j\] and \[((b'_1, i'_1, t'_1)^j', (b'_2, i'_2, t'_2)^j') \in A^{j'}\]

with \(t_1 \leq t'_1\) conflict if either

- \(u^j_{i_1} u^j_{i_2} = u'^{j'}_{i'_1} u'^{j'}_{i'_2} = uv \in A\) and
  \(t_1 + t^H_{uv}(m(j), (b_1, b_2), m(j'), (b'_1, b'_2)) > t'_1\), or
- \(u^j_{i_1} u^j_{i_2} = u'^{j'}_{i'_2} u'^{j'}_{i'_1} = uv \in A_S\) and
  \(t_1 + t^{HS}_{uv}(m(j), (b_1, b_2), m(j'), (b'_1, b'_2)) > t'_1\).

Lead to constraints of the type

\[\sum_{a \in C} x_a \leq 1,\]

where \(C\) is a clique in the conflict graph.
Headway constraints

Example:
• train 1 first, train 2 second: 3 minutes
• train 2 first, train 1 second: 2 minutes

Conflict graph

Constraint

\[ \sum_{a \in \{ \text{red arcs} \}} x_a \leq 1 \]
Objective function

- high costs on infeasible-arcs,
- no costs on running-arcs (running is good),
- increasing costs on waiting arcs (waiting is bad)
ILP formulation

maximize \( \sum_{a \in A} x_a w_a \)

subject to

flow conservation
\[
\begin{align*}
\sum_{a \in \delta^+(\sigma j)} x_a &= 1, & j \in T, \\
\sum_{a \in \delta^+ (v)} x_a &= \sum_{a \in \delta^- (v)} x_a, & j \in T, v \in V^j \setminus \{\sigma^j, \tau^j\}, \\
\sum_{a \in \delta^- (v, t)} x_a &\leq c_v, & v \in V, t \in S,
\end{align*}
\]

capacity
\[
\begin{align*}
\sum_{a \in \delta^- (uv, t)} x_a &\leq c_{uv}, & uv \in A, t \in S,
\end{align*}
\]

headway
\[
\begin{align*}
\sum_{a \in C} x_a &\leq 1, & C \in \mathcal{C},
\end{align*}
\]

binary
\[
\begin{align*}
x_a &\in \{0, 1\}, & a \in A.
\end{align*}
\]
Goal: rounding heuristics based on a relaxation of the ILP.

Because of the large size of the instances, solving the LP relaxation by a state-of-the-art solver is too slow.

⇒ solve the Lagrangian dual obtained by relaxation of the coupling constraints.
Lagrange dual and decomposition

Let

- \( D x \leq d \) be the coupling constraints,
- \( D^j, j \in T \), be the columns corresponding to the \( x_a, a \in A^j \),
- \( X^j = \left\{ x \in \mathbb{R}^{A^j} : x \text{ is valid path in } G^j \right\} \).

The LP reads

\[
\max_{\substack{D x \leq d \\ x \in X}} w^T x
\]

with the Lagrangian dual problem

\[
\inf_{y \geq 0} \left( d^T y + \sum_{j \in T} \max_{x^j \in X^j} \left[ \left( w^j - D^j y \right)^T x^j \right] \right).
\]
Bundle method

The bundle method requires the evaluation of

$$\varphi(y) = d^T y + \sum_{j \in T} \max_{x^j \in \mathcal{X}^j} \left[ (w^j - D^j y)^T x^j \right]$$

for given $y$.

These are \textit{independent shortest-path problems}.

Each optimal solution $x(y)$ of the shortest path problems yields a \textit{subgradient}

$$g(y) = d - Dx(y).$$

The bundle method (see, e.g., Lemaréchal)

- requires an oracle returning the \textit{function value} and a \textit{subgradient},
- generates a sequence of \textit{convex-combinations} of the paths returned by the oracle, the so called \textit{primal aggregates}.


Primal aggregates and separation

The primal aggregates

- converge to an optimal solution of the LP-relaxation of the TTP ⇒ can be used by rounding heuristics,
- may be used for *primal separation* of the conflict constraints, see Helmberg (2004).

Why primal separation?

- capacity constraints: relatively small number, easy to separate,
- headway constraints: possibly exponentially large number, separated by heuristics.
Numerical results

Three test instances based on south-west network of DB (roughly Baden-Wuerttemberg):
Numerical results

Three test instances based on south-west network of DB (roughly Baden-Wuerttemberg):

1. A small part of the network containing the five most frequently used arcs,
2. the main long-distance and freight traffic route along the river Rhine,
3. the whole subnet.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Passenger</th>
<th>Freight</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>104</td>
<td>193</td>
<td>242</td>
<td>9</td>
<td>317336</td>
</tr>
<tr>
<td>2</td>
<td>656</td>
<td>1210</td>
<td>50</td>
<td>67</td>
<td>2448842</td>
</tr>
<tr>
<td>3</td>
<td>2103</td>
<td>4681</td>
<td>2501</td>
<td>659</td>
<td>8990060</td>
</tr>
</tbody>
</table>
Solving the relaxation

Memory and time consumption by CPLEX and ConicBundle (on an Intel Xeon Dual Core, 3 GHz, 16 GB RAM):

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPLEX</th>
<th>ConicBundle</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33s</td>
<td>12s</td>
<td>160 MB</td>
</tr>
<tr>
<td>2</td>
<td>1945s</td>
<td>341s</td>
<td>1 GB</td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>2512s</td>
<td>6 GB</td>
</tr>
</tbody>
</table>

Development of the objective function:
First integer results

First round heuristics based on successive fixation of arcs yielded good results for instance 1 and 2, but not for 3:

<table>
<thead>
<tr>
<th>Instance</th>
<th>Time</th>
<th>Infeasible trains</th>
<th>Late trains</th>
<th>Average delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39s</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>697s</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3182s</td>
<td>40</td>
<td>906</td>
<td>865s</td>
</tr>
<tr>
<td>3b</td>
<td>10h</td>
<td>9</td>
<td>778</td>
<td>603s</td>
</tr>
</tbody>
</table>
Reducing the problem size

**Problem:**
- instances are too large,
- separation of headway constraints too expensive.

**Solution ideas:**
- create train-graphs dynamically,
- instead of separation of headway constraints, model feasible *configurations* by configuration-networks.
Dynamic train-graphs

Most trains only use a small part of their trains:

Idea: create only required parts of the network.
Dynamic train-graphs

Dynamically constructed train-graph:

Remark: The dynamic creation of train-graphs requires an appropriate cost-structure (given in our case).
Goal: Replace headway constraints by configuration networks, that model *feasible* train runs.
Configuration networks: structure

(Borndörfer et al, 2007)

- one *configuration network* for each infrastructure arc,
- train-arcs are activated by the configuration-networks,
Configuration networks: structure

(Borndörfer et al, 2007)

- one *configuration network* for each infrastructure arc,
- train-arcs are activated by the configuration-networks,
Configuration networks: structure

(Borndörfer et al, 2007)

- one configuration network for each infrastructure arc,
- train-arcs are activated by the configuration-networks,

![Train graph and example configuration](image-url)
Configuration networks

Pros:

- no separation of headway constraints necessary,
- instead simple coupling constraints between train-graphs and configuration networks.

Cons:

- number of variables increases a lot,
- dynamic generation of configuration networks required.
Next steps

- implementation of (dynamic) configuration networks,
- exploit dual sensitivity information for better rounding heuristics,
- robustness.
Thank you for your attention.