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

DB Mobility Networks Logistics

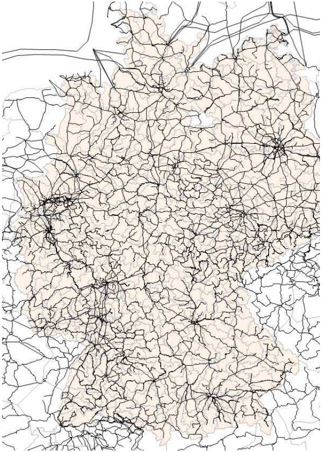
OVERSYS

TECHNISCHE UNIVERSITÄT CHEMNITZ

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

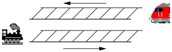
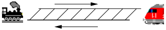
Former work

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)
  - Brännlund, Lindberg, Nõu, Nilsson (1998)
  - Caprara, Fischetti, Toth (2002)
  - Cacchiani, Caprara, Toth (2006)
  - Caprara, Kroon, Monaci, Peeters, Toth (2006)
  - Ingolotti, Baber, Tormos, Lova, Ealido, Abril (2006)
  - Borndörfer, Schlechte (2007)

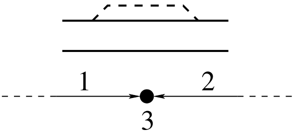
Problem data

Given:

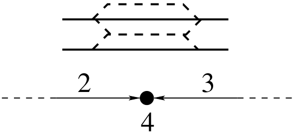
- *infrastructure digraph* $D = (V, A)$ where
  - V set of stations, track switches ...
  - A set of tracks, may be
    - *double tracks* 
    - *single tracks* 
  - *absolute node capacities*
  - *directional capacities*

Problem data

Example: absolute and directional capacities.



absolute	left dir.	right. dir.
3	1	2



absolute	left dir.	right. dir.
4	2	3

Trains

For each train $j \in T$:

- *train type* $m(j) \in M$
- *predefined route:* ordered sequence of nodes $U(j) = (u_1^j, \ldots, u_{n_j}^j), n_j \in \mathbb{N}$

Furthermore for each passenger train

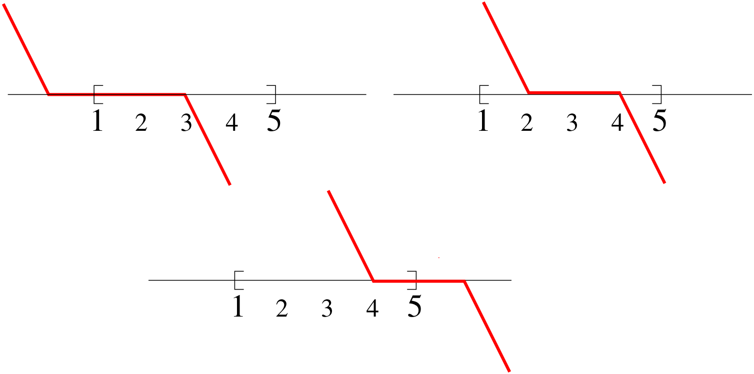
- *stopping interval* $I_i^j = \left[t_i^{S,j}, t_i^{E,j}\right] \subset \mathbb{Z} \cup \{\pm\infty\}$
  "when the train has to wait"
- *minimal stopping time* $d_i^j \in \mathbb{Z}_+$.
  "how long the train has to wait"

train must arrive before $t_i^{E,j}$ and must not leave before $t_i^{S,j} + d_i^j$.

Stopping interval and minimal stopping time

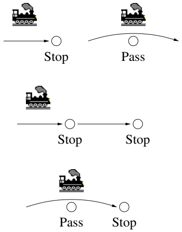
Example: stopping interval $= [1, 5]$, minimal stopping time $= 2$ minutes

The following examples are valid:



Running times

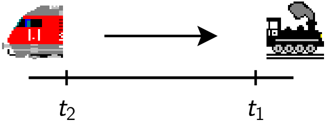
A train needs some time from one station to the next, its *running time*.
This depends on the train type ($m \in M$) and on whether the train *stops* or *passes* at the stations:



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

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:

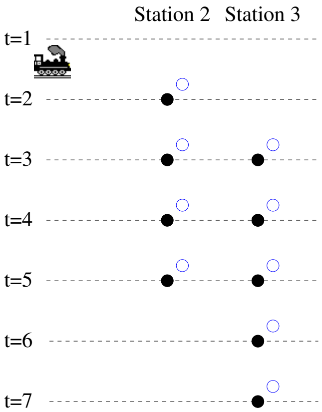
$$t_a^H \colon M \times B_R \times M \times B_R \to \mathbb{Z}_+.$$

Model

Classic model via *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 *start-node* σ^j,
  - an artificial *end-node* τ^j,
  - a *wait-node* and a *stop-node* node for each *station*, *time-step*
- A^j contains
  - *starting arcs* from σ^j to the first station's nodes,
  - *ending arcs* from the last station's nodes to τ^j,
  - *waiting arcs* between two successive wait-nodes of one station,
  - *running arcs* connecting nodes of successive stations
  - *infeasible arcs* from each intermediate station's node to τ^j.

Train graphs: nodes

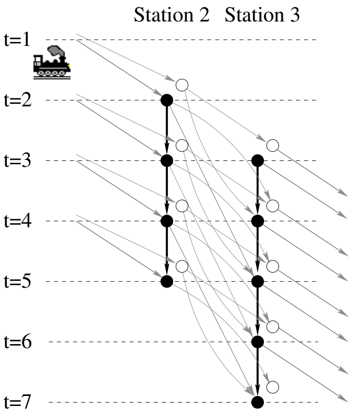


- ● stop nodes
  train *stops* at the station

- ○ run nodes
  train *passes* through the station

nodes

Train graphs: waiting arcs

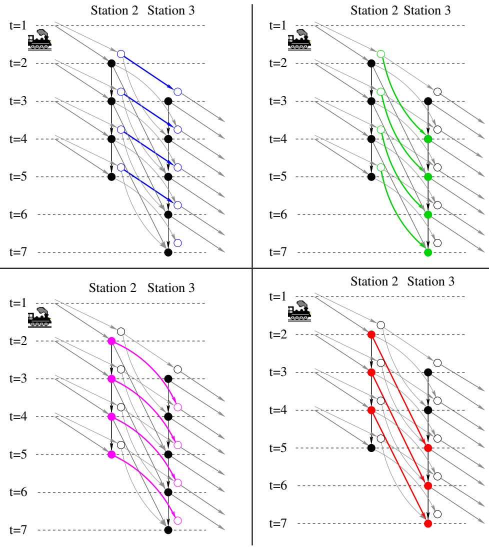


waiting arcs

Train graphs: running arcs

run-
run

run-
stop

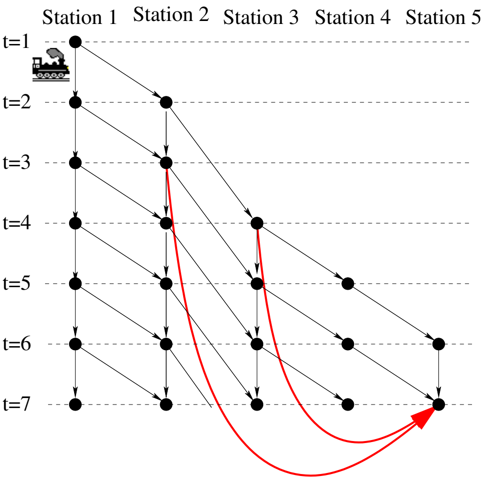


start-
run

start-
stop

running arcs

Train graphs: infeasible arcs



infeasible arcs

Variables

Let $\mathcal{A} := \bigcup_{j \in T} A^j$ 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 $a \in A^j$:

$$x_a = 1 \Leftrightarrow \text{train } j \text{ uses arc } a.$$

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

$$\sum_{a \in \delta^-(v,t)} x_a \le c_v \qquad \text{absolute capacities}$$

and

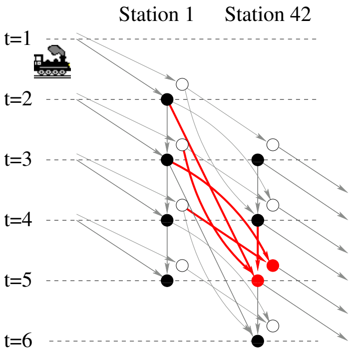
$$\sum_{a \in \delta^-(uv,t)} x_a \le c_{uv}, \qquad \text{directional capacities}$$

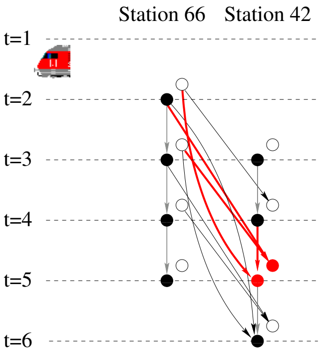
where

$$\delta^-(v,t) = \left\{ ((b',i',t')^j, (b,i,t)^j) \in \mathcal{A} \colon u_i^j = v \right\},$$
$$\delta^-(uv,t) = \left\{ ((b',i',t')^j, (b,i,t)^j) \in \mathcal{A} \colon u_{i-1}^j u_i^j = uv \right\}.$$

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

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

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

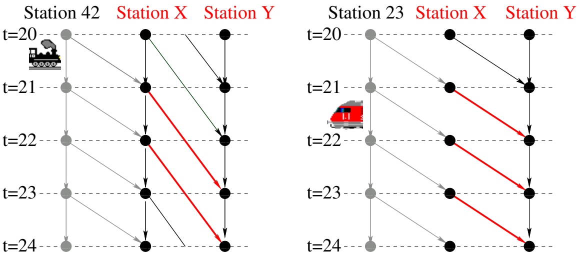
$$\sum_{a \in C} x_a \leq 1,$$

where C is a clique in the conflict graph.

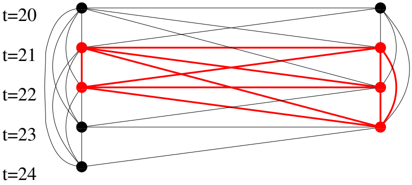
Headway constraints

Example:

- train 1 first, train 2 second: 3 minutes
- train 2 first, train 1 second: 2 minutes



Conflict graph

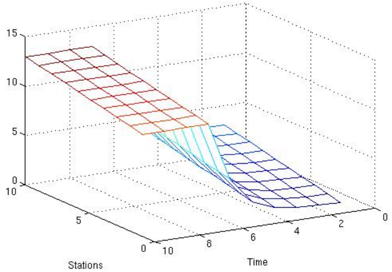


Constraint

$$\sum_{a \in \{\text{red arcs}\}} x_a \leq 1$$

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

$$\text{maximize} \sum_{a \in \mathcal{A}} x_a w_a$$

subject to

$$\text{flow conservation} \begin{cases} \displaystyle\sum_{a \in \delta^+(\sigma^j)} x_a = 1, & j \in T, \\ \displaystyle\sum_{a \in \delta^+(v)} x_a = \sum_{a \in \delta^-(v)} x_a, & j \in T, v \in V^j \setminus \{\sigma^j, \tau^j\}, \end{cases}$$

$$\text{capacity} \begin{cases} \displaystyle\sum_{a \in \delta^-(v,t)} x_a \leq c_v, & v \in V, t \in S, \\ \displaystyle\sum_{a \in \delta^-(uv,t)} x_a \leq c_{uv}, & uv \in A, t \in S, \end{cases}$$

$$\text{headway} \begin{cases} \displaystyle\sum_{a \in C} x_a \leq 1, & C \in \mathcal{C}, \end{cases}$$

$$\text{binary} \begin{cases} x_a \in \{0, 1\}, & a \in \mathcal{A}. \end{cases}$$

Solution methods

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.

$\Rightarrow$ solve the Lagrangian dual obtained by relaxation of the coupling constraints.

Lagrange dual and decomposition

Let

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

The LP reads

$$\max_{\substack{Dx \leq d \\ x \in \mathcal{X}}} w^T x$$

with the Lagrangian dual problem

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

Bundle method

The bundle method requires the evaluation of

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

for given y.
These are *independent shortest-path problems*.
Each optimal solution $x(y)$ of the shortest path problems yields a *subgradient*

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

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

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

Primal aggregates and separation

The primal aggregates

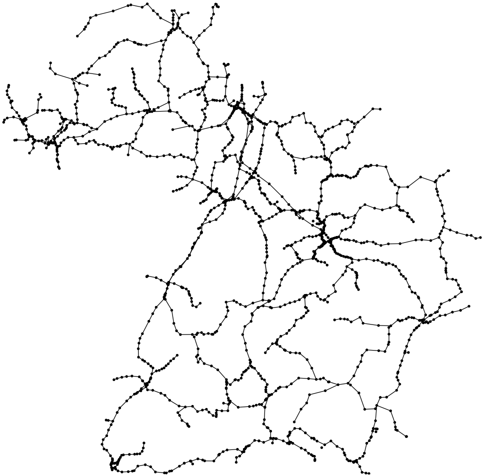
- converge to an optimal solution of the LP-relaxation of the TTP $\Rightarrow$ can be used by rounding heuristics,
- may be used for *primal separation* of the conflict constraints, see Helmberg (2004).

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.

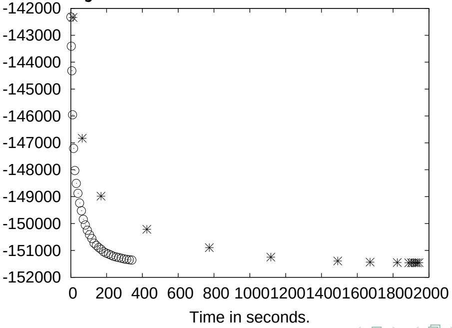
Instance	Nodes	Arcs	Passenger	Freight	Variables
1	104	193	242	9	317336
2	656	1210	50	67	2448842
3	2103	4681	2501	659	8990060

Solving the relaxation

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

Instance	CPLEX	ConicBundle	Size
1	33s	12s	160 MB
2	1945s	341s	1 GB
3	–	2512s	6 GB

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:

Instance	Time	Infeasible trains	Late trains	Average delays
1	39s	0	0	0
2	697s	0	0	0
3	3182s	40	906	865s
3b	10h	9	778	603s

Reducing the problem size

Problem:

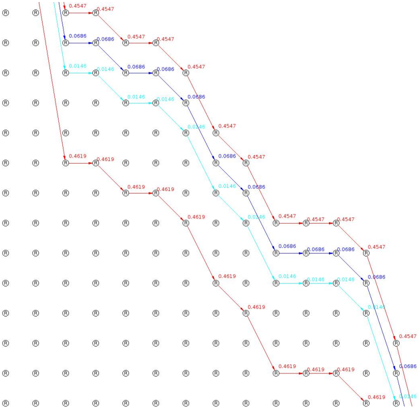
- instances are too large,
- separation of headway constraints too expensive.

Solution ideas:

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

Dynamic train-graphs

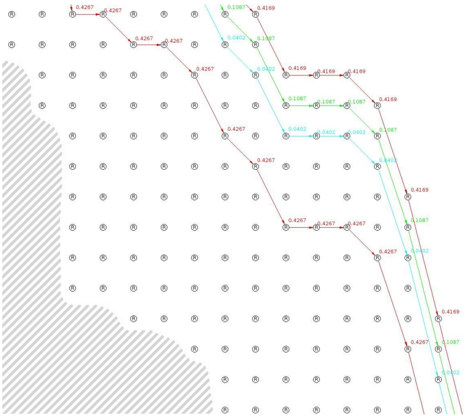
Most trains only use a small part of their trains:



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).

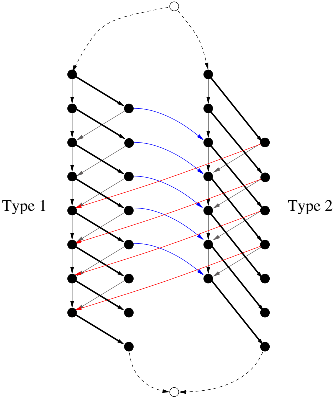
Configuration networks

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,

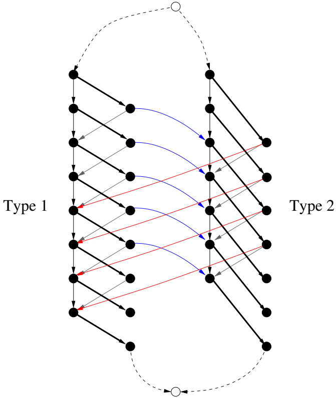


configuration graph

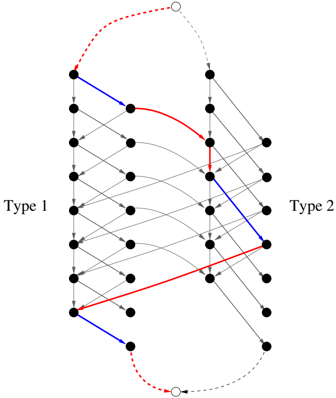
Configuration networks: structure

(Borndörfer et al, 2007)

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



configuration graph

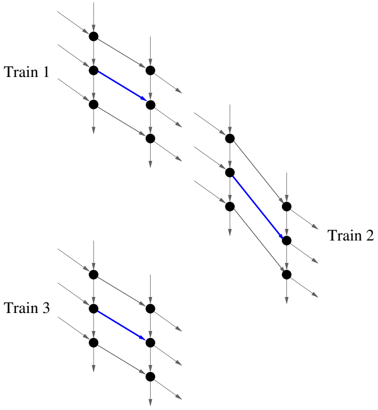


example configuration

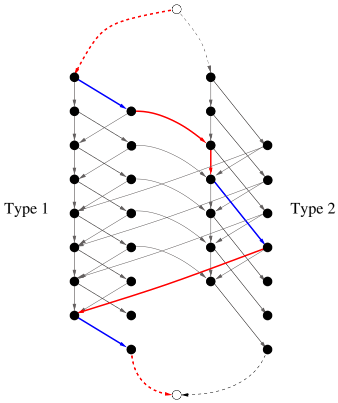
Configuration networks: structure

(Borndörfer et al, 2007)

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



train graph



example configuration

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.