Robust Train Timetabling

Matteo Fischetti – DEI, University of Padova
Domenico Salvagnin – DMPA, University of Padova
Arrigo Zanette – DMPA, University of Padova

ATMOS 2007, Sevilla, November 15–16 2007

Work supported by EU (project ARRIVAL) and MIUR (PRIN 2006)
Problem definition

- Single one-way line
- Aperiodic daily timetable to be designed
- Minimize the timetable cost computed as
An event scheduling MIP model

Variables:
- Arrival and departure times \( t_i \)
- Binary variables modeling events precedences \( x_{ij} \)

Constraints:
- Minimum travel times: \( d_{ij} \)
- Safety constraints
  headway times, no overtaking, etc.
- Typical constraints: \( t_i - t_j \geq d_{ij} - Mx_{ij} \)

Objectives:
- Minimize the cost of the schedule
- Robustness (whatever it means)
Timetable robustness . . .

- . . . not concerned with major disruptions

- . . . is not intended to cope with heavy truck breaks or alike
  - to be handled by the REAL TIME CONTROL SYSTEM

- . . . is a way to control delay propagation
- . . . has to favor delay compensation without heavy actions from the traffic control center

- no overtaking allowed to prevent delay propagation
- no train cancellation
- train precedences unchanged w.r.t the planned timetable
Our approach

- Take a feasible timetable
  - near optimal solution of the “nominal” timetable problem
- Fix a maximum price of robustness
  - the cost of the robust solution cannot exceed by more than XX% the optimal cost of the nominal problem
- Fix all train precedences (binary var.s $x_{ij}$ in the MIP model)
- Relax the integrality on the event-time var.s $t_i$ (the only unknowns)
- Enforce robustness in the resulting LP using alternative techniques
- Evaluate the achieved robustness through a common validation model
- Compare the results
Pursuing robustness in the LP/MIP context

Stochastic Programming

- Take first-stage decisions
- Pay for restoring feasibility afterwards (second-stage recourse var.s)
Pursuing robustness in the LP/MIP context

Stochastic Programming

- Take first-stage decisions
- Pay for restoring feasibility afterwards (second-stage recourse var.s)

- Very flexible: applied successfully by the Kroon’s group to periodic timetabling
- Computationally heavy in scenario-based approaches
Pursuing robustness in the LP/MIP context

Robustness à la Bertsimas-Sim

- Kind of worst-case analysis of robustness
- Limits the moves of the adversary (just a few coefficients can change in each constraints)
- Feasibility deterministic (if adversary behaves as expected) or with high probability (otherwise)
Pursuing robustness in the LP/MIP context

Robustness à la Bertsimas-Sim

- Kind of worst-case analysis of robustness
- Limits the moves of the adversary (just a few coefficients can change in each constraints)
- Feasibility deterministic (if adversary behaves as expected) or with high probability (otherwise)
- Very simple model
- Unfortunately, of no use in the timetable context (infeasible or very inefficient solutions)
Pursuing robustness in the LP/MIP context

Light Robustness

- “Light” version of Bertsimas-Sim using slack variables for “too conservative” constr.s
- Focus on slack var.s: put knowledge about uncertainty on their bounds
- Minimize some weight function of slack variables
The validation model

Simulation tool used to evaluate the actual robustness of a given timetable \((\tilde{x}, \tilde{t})\)

- Uses information on the line to generate a delay scenario for each run
- For each run, solves an LP model to absorb as much delay as possible:
  - Fixed precedences
    - \(x_{ij} \coloneqq \tilde{x}_{ij}\), continuous event-time var.s \(t_i\) only
  - Cannot anticipate with respect to the input solution:
    - actual times in the delayed schedule are \(t_i \geq \tilde{t}_i\)
  - Minimize the cumulative delay:
    - \(\min \sum_i (t_i - \tilde{t}_i)\)
- Gather statistical information about the cumulative delays
The “fat” stochastic programming model

- 2-stage SP model with tradeoff constraint
  - cost(t) ≤ (1 + α)cost(t*)
- Fixed precedences
  - x_{ij} := \tilde{x}_{ij}, continuous event-time var.s t_i only
- Recourse variables \( \hat{t}_i^\omega \) for each scenario \( \omega \)
  - complete duplication of the nominal set of var.s \( t_i \) and constr.s
- Cannot anticipate with respect to the input solution:
  - \( \hat{t}_i^\omega \geq t_i \)
- Minimize the expected cumulative delay:
  - \( \min \mathbb{E} \sum_i (\hat{t}_i^\omega - t_i) \)
The “fat” stochastic programming model

- 2-stage SP model with tradeoff constraint
  - \( \text{cost}(t) \leq (1 + \alpha)\text{cost}(t^*) \)
- Fixed precedences
  - \( x_{ij} := \tilde{x}_{ij} \), continuous event-time var.s \( t_i \) only
- Recourse variables \( \hat{t}_i^\omega \) for each scenario \( \omega \)
  - Complete duplication of the nominal set of var.s \( t_i \) and constr.s
- Cannot anticipate with respect to the input solution:
  - \( \hat{t}_i^\omega \geq t_i \)
- Minimize the expected cumulative delay:
  - \( \min \mathbb{E} \sum_i (\hat{t}_i^\omega - t_i) \)

Much like the validator model.
The “Slim v.1” Stochastic Programming Model

- 2-stage SP model with tradeoff constraint
- Fixed precedences
  - \( x_{ij} := \tilde{x}_{ij} \), continuous event-time var.s \( t_i \) only
- Recourse variables are slacks \( s_{ij}^{\omega} \) (i.e. unabsorbed extra-time) on original constr.s:
  - \( t_i - t_j + s_{ij}^{\omega} \geq d_{ij} + \delta_{ij}^{\omega} \) → one for each constraint where \( \delta_{ij}^{\omega} \neq 0 \)
- Minimize the expected sum of slacks:
  - \( \min \mathbb{E} \sum_{ij} s_{ij}^{\omega} \)
The “Slim v.1” Stochastic Programming Model

- 2-stage SP model with tradeoff constraint
- Fixed precedences
  - $x_{ij} := \tilde{x}_{ij}$, continuous event-time var.s $t_i$ only
- Recourse variables are slacks $s_{ij}^\omega$ (i.e. unabsorbed extra-time) on original constr.s:
  - $t_i - t_j + s_{ij}^\omega \geq d_{ij} + \delta_{ij}^\omega \rightarrow$ one for each constraint where $\delta_{ij}^\omega \neq 0$
- Minimize the expected sum of slacks:
  - $\min \mathbb{E} \sum_{ij} s_{ij}^\omega$

Uniformly pay for unabsorbed delay along the line
The “Slim v.2” Stochastic Programming Model

But early delays propagate to subsequent line sections.
But early delays propagate to subsequent line sections.

- Minimize the expected propagated delay:
  - assume the unabsorbed delay stay unabsorbed till the end of the line
  - each slack is weighted by $w_{ij}$, the number of subsequent line segments:
    \[
    \min \mathbb{E} \sum_{ij} w_{ij} s_{ij}^\omega
    \]
A single slack $s_{ij}$ per constraint

A bound $s_{ij} \leq \beta_{\text{max}}$ ensuring the maximum protection level

eg. $\beta_{\text{max}}$ such that $\Pr[\delta_{ij}^\omega \leq \beta_{\text{max}}] = 0.5$

$\min \mathbb{E} \sum_{ij} w_{ij} s_{ij}^\omega$

(*) Less than 1 calorie MByte per serving
Computational results

- Stochastic model solved by means of Sample Average Approximation (SAA)
  - $\omega \rightarrow r$, finite number of $N$ samples
  - $\mathbb{E}[x^\omega] \rightarrow \sum_{1}^{N} 1/N \times r$
- Latin-Hypercube variance reduction sampling
- Exponentially distributed ($\mu = 5$) cumulative train delays
Testbed

- Real world instances from RFI:
  - PD-BO: 17 stations, \( \sim 35 \) trains
  - BZ-VR: 27 stations, \( \sim 130 \) trains
  - Mu-VR: 48 stations, \( \sim 50 \) trains
  - Br-BO: 48 stations, \( \sim 70 \) trains

- For each instance, 5 almost-optimal (non-robust) timetables computed by DEIS, University of Bologna
Validation results

Line BrBO

Cumulative delay (min)

Efficiency loss

fat
slim 1
slim 2
LR

M. Fischetti, D. Salvagnin, A. Zanette, Robust Train Timetabling
Computing times

- Computing the robust solution of BZ-VR:
  - takes $\sim 10000$ seconds using the fat model with 50 scenarios
  - takes $\sim 1000$ seconds using the slim1 and slim2 models
  - takes $\sim 10$ seconds using the light robustness model

"fat" is significantly slower than other methods
"slim2 v.2" is not significantly slower than "slim v.1"
"LR" is definitely the fastest
Computing times

- Computing the robust solution of BZ-VR:
  - takes $\sim 10000$ seconds using the fat model with 50 scenarios
  - takes $\sim 1000$ seconds using the slim1 and slim2 models
  - takes $\sim 10$ seconds using the light robustness model

- “fat” is significantly slower than other methods
- “slim2 v.2” is not significantly slower than “slim v.1”
- “LR” is definitely the fastest
The structure of time buffers

- A nice figure of the way buffers are allocated is their Weighted Average Distance (WAD) from the starting point.
  - WAD = 0.5, buffers are proportionally allocated along the line.
  - WAD > 0.5, buffers are mainly allocated in the second part.
  - WAD < 0.5, buffers are mainly allocated in the first part.
Buffers on an instance

Buffer allocation curves in MUVR (eff.loss. 10%)

M. Fischetti, D. Salvagnin, A. Zanette, Robust Train Timetabling
That’s all!
Thank you for your attention