Robustness in Railway Operations

Richard Lusby

Department of Management Engineering
Technical University of Denmark

Railway Operations Research Seminar
December 10th, 2014
Agenda

1. Introduction
- Robustness/RobustRailS
- Danish State Railways (DSB)

2. Case Studies
- Rolling Stock Rescheduling
- Depot Planning Integration
- Independent Rail Systems

3. Conclusions & Comments
Punctuality, reliability, recoverability are key performance measures within the railway industry.

All are related to the robustness of the service.

How resistant is the system to minor disturbances?

How recoverable is the system if it does “break”? 

There is an increasing need for fast, reliable decision support systems:
- bigger problems
- in some cases solutions are needed “instantly”
- assurance things are being done as well as can be

Improving the quality of the Danish rail system given high priority
- Research Project: RobustRailS
RobustRailS

- Funded by the Danish Council for Strategic Research
- Large interdisciplinary project tackling railway robustness
  - DTU Transport, DTU Fotonik, and DTU Compute
- In collaboration with all large rail stakeholders in Denmark
  - DSB, DSB S-tog, BaneDanmark, and the Danish Transport Authority
- Holistic approach to improving the reliability of the Danish Railways
- Topics considered (among others)
  - Formal Development and Verification of Railway Control Systems
  - Communication Technologies Support
  - Robust Planning
  - Integrated Disruption Management
  - Passenger Behaviour Models
Robustness in Railway Operations

RobustRails is a large interdisciplinary project that will attempt to answer the question that is constantly being asked in the media and among commuters, as well as by politicians and the operators themselves: Can we get the trains to run on time?

This project is funded by the Danish Council for Strategic Research and runs for the four year period 2012-2015. Read more about the project here.

IC3 Train (Inter-City Train)

www.robustrails.man.dtu.dk
Case Study 1:
Rolling Stock Rescheduling at DSB-Stog

Jørgen Haahr    Richard Lusby    David Pisinger    Jesper Larsen
Overview

- Things do not always go to plan on the day of operation
- Unforeseen disturbances disrupt planned schedules
  - Introduces infeasibilities
- Many interdependent problems need simultaneous optimization
- Here the focus is rolling stock rescheduling (→ depot planning)
- Re-allocate units to substrips to provide sufficient seat coverage
  - Assume a revised timetable is available
  - Reschedule units as best as possible
  - A number of soft constraints
  - Hard constraint on depot capacities
- Tight time limit on finding a solution
- Currently done manually ...
Rolling Stock (Re)-scheduling

Richard Lusby (DTU Management Eng.) Robustness in Railway Operations 10/12/2014 9/35
Rolling Stock (Re)-scheduling
Two different unit types (SE, SA)

- Can be combined in different ways giving different compositions
- \( \approx 130 \) train units in total, \( \approx 1100 \) subtrips, \( \approx 29000 \) stops per day
- At most 10 active depots
- Ideally want to be able to model individual unit movements
  - Gives more flexibility when enforcing maintenance requirements
- Propose a path based Mixed Integer Programming (MIP) formulation
Solving the Rolling Stock Scheduling Problem
Mathematical Formulation

\[ \sum_{p \in \mathcal{P}_d^u} \lambda_p = inv_d^u \quad \forall u \in \mathcal{U}, d \in \mathcal{D} \]

\[ \sum_{p \in \mathcal{P}} \alpha^t_p \lambda_p \geq 1 - y_t \quad \forall t \in \mathcal{T} \]

\[ \sum_{u \in \mathcal{U}} s_u \sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d^u} \alpha^t_p \lambda_p \geq demand_t - z_t \quad \forall t \in \mathcal{T} \]

\[ \sum_{p \in \mathcal{P}} \beta^d_p \lambda_p \geq eod_d^u - w_d^u \quad \forall u \in \mathcal{U}, d \in \mathcal{D} \]

\[ \sum_{u \in \mathcal{U}} \sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d^u} \alpha^t_p \lambda_p \leq length_t \quad \forall t \in \mathcal{T} \]

\[ \sum_{u \in \mathcal{U}} \sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d^u} \gamma^{\text{depot}(a)}_p \lambda_p \leq track_{\text{depot}(a)} \quad \forall a \in \mathcal{A} \]

\[ \lambda_p \in \{0, 1\} \quad y_t \in [0; 1] \quad z_t, w_d^u \in \mathbb{Z}_0^+ \]
Solution Method Details

- Unit based path model (anonymous routes)
- Solved using dynamic column generation
- Subproblem: shortest path problem in a time space network
  - Price all depot × unit-types
- Complete Branch-and-Price framework enforces integrality
Subproblem - Example
### Scenario - Central Segment Blocked

<table>
<thead>
<tr>
<th>Contingency Plan</th>
<th>A.0.16 (København H - Østerport)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>One track is blocked in the central segment</td>
</tr>
<tr>
<td>Cancelled lines</td>
<td>A2, B2, Bx, C1, E2, H</td>
</tr>
<tr>
<td>Turned lines</td>
<td>A1, B1, C2, A1, C1</td>
</tr>
<tr>
<td>Unchanged</td>
<td>E1</td>
</tr>
</tbody>
</table>
Case Study:

<table>
<thead>
<tr>
<th>Disruption</th>
<th>#</th>
<th>Time</th>
<th>Cols</th>
<th>Gap</th>
<th>Root</th>
<th>Cover</th>
<th>Seat</th>
<th>Mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fri 9:00-10:00</td>
<td>836</td>
<td>11</td>
<td>4 096</td>
<td>0.3%</td>
<td>15.1%</td>
<td>99.8%</td>
<td>98.9%</td>
<td>122.4%</td>
</tr>
<tr>
<td>Fri 9:00-13:00</td>
<td>782</td>
<td>18</td>
<td>6 068</td>
<td>0.5%</td>
<td>14.7%</td>
<td>99.8%</td>
<td>98.7%</td>
<td>119.4%</td>
</tr>
<tr>
<td>Fri 11:00-12:00</td>
<td>713</td>
<td>8</td>
<td>3 051</td>
<td>0.6%</td>
<td>16.7%</td>
<td>99.8%</td>
<td>98.9%</td>
<td>122.1%</td>
</tr>
<tr>
<td>Fri 11:00-15:00</td>
<td>657</td>
<td>10</td>
<td>3 939</td>
<td>0.2%</td>
<td>14.8%</td>
<td>99.7%</td>
<td>98.4%</td>
<td>117.5%</td>
</tr>
<tr>
<td>Fri 15:00-16:00</td>
<td>468</td>
<td>6</td>
<td>2 379</td>
<td>0.3%</td>
<td>2.1%</td>
<td>99.6%</td>
<td>97.2%</td>
<td>113.8%</td>
</tr>
<tr>
<td>Fri 15:00-19:00</td>
<td>402</td>
<td>5</td>
<td>1 727</td>
<td>0.0%</td>
<td>1.9%</td>
<td>99.8%</td>
<td>98.4%</td>
<td>115.3%</td>
</tr>
<tr>
<td>Mon 9:00-10:00</td>
<td>820</td>
<td>13</td>
<td>4 129</td>
<td>0.4%</td>
<td>2.7%</td>
<td>99.8%</td>
<td>98.4%</td>
<td>121.6%</td>
</tr>
<tr>
<td>Mon 9:00-13:00</td>
<td>766</td>
<td>14</td>
<td>5 827</td>
<td>0.1%</td>
<td>2.5%</td>
<td>99.8%</td>
<td>98.8%</td>
<td>118.3%</td>
</tr>
<tr>
<td>Mon 11:00-12:00</td>
<td>697</td>
<td>10</td>
<td>3 665</td>
<td>0.5%</td>
<td>16.5%</td>
<td>99.8%</td>
<td>98.8%</td>
<td>122.4%</td>
</tr>
<tr>
<td>Mon 11:00-15:00</td>
<td>641</td>
<td>8</td>
<td>4 066</td>
<td>0.2%</td>
<td>14.3%</td>
<td>99.7%</td>
<td>98.4%</td>
<td>118.6%</td>
</tr>
<tr>
<td>Mon 15:00-16:00</td>
<td>451</td>
<td>6</td>
<td>2 239</td>
<td>0.0%</td>
<td>1.5%</td>
<td>99.6%</td>
<td>97.5%</td>
<td>119.1%</td>
</tr>
<tr>
<td>Mon 15:00-19:00</td>
<td>385</td>
<td>6</td>
<td>2 053</td>
<td>0.2%</td>
<td>1.9%</td>
<td>99.8%</td>
<td>97.2%</td>
<td>118.2%</td>
</tr>
</tbody>
</table>

System: Ubuntu 13.10, GCC 4.8.1, Machine: Intel(R) Xeon(R) CPU X5550 @ 2.67GHz, with 24GB ram
Summary

- Efficient algorithm for rolling stock rescheduling
- Can be easily extended to include compositions
- Framework should be extended to include maintenance requirements
Case Study 2:

Integrating Depot Planning at DSB-Stog

Richard Lusby    Jørgen Haahr    Jesper Larsen    David Pisinger
Overview

- Concerns train units not in active circulation
  - Where, what track, and in which position?
- Not well studied
  - Even less so when it comes to routing integration
- Typically enforced with an aggregated constraint on total length
- Train ordering on depot tracks is not modelled
- Units typically accessed in LIFO fashion
  - All S-tog’s depots are LIFO stacks
- Critical problem when depot capacity is scarce
- Can we extend the rolling stock framework to include depot planning?
Overview

- Concerns train units not in active circulation
  - Where, what track, and in which position?
- Not well studied
  - Even less so when it comes to routing integration
- Typically enforced with an aggregated constraint on total length
- Train ordering on depot tracks is not modelled
- Units typically accessed in LIFO fashion
  - All S-tog’s depots are LIFO stacks
- Critical problem when depot capacity is scarce

Can we extend the rolling stock framework to include depot planning?
Approach

- Rolling Stock framework provides \texttt{anonymous} unit routes
- Solution satisfies depot stock levels and aggregated capacity
- Could be \texttt{infeasible} with respect to parking order
- It is possible to \texttt{test} routing solutions for parking feasibility
- Produce a separation routine to cut \texttt{parking infeasible} solutions away
- Extend routing framework to \texttt{Branch-and-Price-and-Cut}
Approach

- Rolling Stock framework provides anonymous unit routes
- Solution satisfies depot stock levels and aggregated capacity
- Could be infeasible with respect to parking order
- It is possible to test routing solutions for parking feasibility
- Produce a separation routine to cut parking infeasible solutions away
- Extend routing framework to Branch-and-Price-and-Cut
The Separation Routine

- Must be solved on finding a routing feasible solution
- From the anonymous unit routes assign physical numbers
  1. Done using a Mixed Integer Program (feasibility)
  2. Assigns routes to physical units
  3. Ensures units can leave initial depot feasibly
- From physical routes construct all depot movements
- For each depot $d \in \mathcal{D}$ solve the Depot Planning Problem

Depot Planning Problem

Given a set of events $\mathcal{E}$ and a set of tracks $\mathcal{T}$, each with a specified length $l_t$, find an assignment of events to tracks such that the track capacity is not violated at any point in time, and LIFO ordering is respected.
The Separation Routine

- Must be solved on finding a routing feasible solution
- From the anonymous unit routes assign physical numbers
  1. Done using a Mixed Integer Program (feasibility)
  2. Assigns routes to physical units
  3. Ensures units can leave initial depot feasibly

- From physical routes construct all depot movements
- For each depot \( d \in \mathcal{D} \) solve the Depot Planning Problem

**Depot Planning Problem**

Given a set of events \( \mathcal{E} \) and a set of tracks \( \mathcal{T} \), each with a specified length \( l_t \), find an assignment of events to tracks such that the track capacity is not violated at any point in time, and LIFO ordering is respected.
Depot Planning Example

1

13:00

2

11:20

3

11:20

4

14:00

Richard Lusby (DTU Management Eng.) Robustness in Railway Operations 10/12/2014
Depot Planning Example

13:00 11:20 11:20
15:00

Richard Lusby (DTU Management Eng.) Robustness in Railway Operations 10/12/2014 20/35
Depot Planning Example
Depot Planning Example

13:00 11:20 11:20
14:00
15:00

Richard Lusby (DTU Management Eng.) Robustness in Railway Operations 10/12/2014 20/35
Solving the Depot Planning Problem
Mathematical Formulations

Minimize: \( \sum_{e \in E} \sum_{t \in T} c_{et} x_{et} + M \sum_{e \in E} y_e \)

subject to:

- \( \sum_{t \in T} x_{et} + y_e = 1 \quad \forall e \in E, \)
- \( \sum_{e \in E_a} l_e x_{et} \leq L_t \quad \forall a \in A, \)
- \( x_{et} + x_{e't} \leq 1 \quad \forall (e, e') \in C, t \in T, \)
- \( x_{et} \in \{0, 1\} \quad \forall e \in E, t \in T, \)
- \( y_e \in \mathbb{R}^+ \quad \forall e \in E. \)

Minimize: \( \sum_{t \in T} \sum_{p \in P_t} c_{tp} x_{tp} + M \sum_{e \in E} y_e \)

subject to:

- \( \sum_{p \in P_t} x_{tp} = 1 \quad \forall t \in T, \)
- \( \sum_{t \in T} \sum_{p \in P_t} \alpha_{ep} x_{tp} + y_e = 1 \quad \forall e \in E, \)
- \( x_{tp} \in \{0, 1\} \quad \forall t \in T, p \in P_t, \)
- \( y_e \in \mathbb{R}^+ \quad \forall e \in E. \)
Solving the Depot Planning Problem
Mathematical Formulations

Minimize: \( \sum_{e \in E} \sum_{t \in T} c_{et} x_{et} + M \sum_{e \in E} y_e \)

Minimize: \( \sum_{t \in T} \sum_{p \in P_t} c_{tp} x_{tp} + M \sum_{e \in E} y_e \)

\( \sum_{t \in T} x_{et} + y_e = 1 \quad \forall e \in E, \)

\( \sum_{t \in T} l_e x_{et} \leq L_t \quad \forall a \in A, \)

\( \sum_{t \in T} \sum_{p \in P_t} \alpha_{ep} x_{tp} + y_e = 1 \quad \forall e \in E, \)

\( x_{et} + x_{e't} \leq 1 \quad \forall (e, e') \in C, t \in T, \)

\( y_e \in \mathbb{R}^+ \quad \forall e \in E. \)

- Solved using delayed column generation
- Columns correspond to parking plans
- **Subproblem:** Resource Constrained Shortest Path problem
- Full Branch-and-Price Framework for enforcing integrality
## Instance Overview

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Stops</th>
<th>Subtrips</th>
<th>Active Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-03-07</td>
<td>Fri</td>
<td>28,719</td>
<td>1086</td>
<td>A, B, Bx, C, E, F &amp; H</td>
</tr>
<tr>
<td>2014-03-08</td>
<td>Sat</td>
<td>20,474</td>
<td>706</td>
<td>A, B, C &amp; F</td>
</tr>
<tr>
<td>2014-03-09</td>
<td>Sun</td>
<td>19,919</td>
<td>690</td>
<td>A, B, C &amp; F</td>
</tr>
<tr>
<td>2014-03-10</td>
<td>Mon</td>
<td>28,017</td>
<td>1074</td>
<td>A, B, Bx, C, E, F &amp; H</td>
</tr>
</tbody>
</table>

- **MIP Approach**: Gurobi 5.6
- **Column Generation**: Coin-OR BCP 1.3.8, Cplex 12.5
## Results

### Comparison

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Gap (%)</th>
<th>Cols.</th>
<th>t (s)</th>
<th>Nodes</th>
<th>Gap (%)</th>
<th>t (s)</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-03-07</td>
<td>Fri</td>
<td>0.17</td>
<td>5431.90</td>
<td>13.83</td>
<td>4.80</td>
<td>0.80</td>
<td>0.069</td>
<td>60.3</td>
</tr>
<tr>
<td>2014-03-08</td>
<td>Sat</td>
<td>0.00</td>
<td>1779.67</td>
<td>0.81</td>
<td>1.33</td>
<td>0.02</td>
<td>0.017</td>
<td>0.00</td>
</tr>
<tr>
<td>2014-03-09</td>
<td>Sun</td>
<td>0.00</td>
<td>891.50</td>
<td>1.50</td>
<td>1.67</td>
<td>0.02</td>
<td>0.013</td>
<td>0.00</td>
</tr>
<tr>
<td>2014-03-10</td>
<td>Mon</td>
<td>0.00</td>
<td>3485.30</td>
<td>7.13</td>
<td>1.60</td>
<td>1.09</td>
<td>0.020</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Gap (%)</th>
<th>Cols.</th>
<th>t (s)</th>
<th>Nodes</th>
<th>Gap (%)</th>
<th>t (s)</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-03-07</td>
<td>Fri</td>
<td>1.67</td>
<td>13223</td>
<td>60.36</td>
<td>31</td>
<td>5.47</td>
<td>0.54</td>
<td>603</td>
</tr>
<tr>
<td>2014-03-08</td>
<td>Sat</td>
<td>0.00</td>
<td>5572</td>
<td>1.53</td>
<td>3</td>
<td>0.12</td>
<td>0.04</td>
<td>0</td>
</tr>
<tr>
<td>2014-03-09</td>
<td>Sun</td>
<td>0.00</td>
<td>2268</td>
<td>5.15</td>
<td>5</td>
<td>0.11</td>
<td>0.04</td>
<td>0</td>
</tr>
<tr>
<td>2014-03-10</td>
<td>Mon</td>
<td>0.00</td>
<td>11096</td>
<td>32.68</td>
<td>5</td>
<td>6.91</td>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

System: Ubuntu 13.10, GCC 4.8.1, Machine: Intel(R) Xeon(R) CPU X5550 @ 2.67GHz, with 24GB ram
Summary

- Comparison of two approaches for depot planning
- MIP performs better in an integrated rescheduling setting
- More testing to be carried out
- With some modification framework can be used in planning stages
Case Study 3:

Independent Rail Subsystems

Simon Bull    Richard Lusby    Jesper Larsen
The focus of this project is on robust planning

- Proactive approach to building delay resistant rolling stock schedules
- Partition rolling stock units into independent subsystems
- Prevent delay propagation across subsystems
  - Make it possible to cancel a line/groups of lines in isolation

- How to choose a good partition?
- How to quantify the benefit of partitioning?
Overview

- The focus of this project is on robust planning
- Proactive approach to building delay resistant rolling stock schedules
- Partition rolling stock units into independent subsystems
- Prevent delay propagation across subsystems
  - Make it possible to cancel a line/groups of lines in isolation

- How to choose a **good** partition?
- How to quantify the **benefit** of partitioning?
Independent Subsystems - Example
Independent Subsystems - Example
Independent Subsystems - Example
Independent Subsystems - Details

- Solved using a flow model
- Exact unit routes are not determined
  - Seek an indication of partition dependence only
- Passenger demand and unit type limits imposed
- So too flow conservation and station capacities
- Ideally want an optimization-based method for generating partitions
DSB Case Study

- Consider a weekly timetable
- \( \approx 12,000 \) subtrips, \( \approx 385 \) units, \( \approx 13 \) unit types
- Analyse approach by creating a number of 2-partition cases
- Cost of a partition relative to unpartitioned case
- **Partitioned Cost:** Minimize the sum of increases in units for all unit types compared with unpartitioned case
- Quantify reduction in potential “interaction”
  - Count lines that share a station and have overlapping compatible units
  - **Partitioned Benefit:** Count how many such lines are removed
- Might be better consider swapping unit count, or number of cancellable lines
14 Different Partitions
Partitions (Impact on Required Units)

Least

Greatest
Summary

- Partitioning can reduce potential unit interaction
  - Increase in the number of units
- Sensitive to chosen partitioning
- Difficult to estimate the actual benefit
  - Route units within a partition
  - Couple with simulation tool
Conclusions

- Presented three different robustness related projects
- Consider both proactive and reactive tools
- Rescheduling framework is nearing completion
- Open questions on independent rail subsystems
Current Projects/Extensions

- Robust Line Planning
  - Incorporate both rail company and passenger preferences
- Track maintenance scheduling
- Extensions to rescheduling framework (crew, timetable)
- Acceleration techniques
Thank you

Questions or comments?