A Column Generation based Heuristic for Train Driver Rescheduling

Daniel Potthoff
potthoff@few.eur.nl

joint work with D. Huisman and G. Desaulniers

June 18, 2008
1 Introduction

2 Model

3 Algorithm

4 Computational results

5 Future research
Introduction

Reasons for unexpected disruptions

- Infrastructure malfunctions
  - Rails, switches, catenary, bridges
- Computer problems in control centers
- Rolling stock breakdowns
- Accidents with other traffic
- Weather conditions
- Crew no shows
- ...

<table>
<thead>
<tr>
<th>Disruptions</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>933</td>
</tr>
<tr>
<td>Medium</td>
<td>1011</td>
</tr>
<tr>
<td>Large</td>
<td>834</td>
</tr>
</tbody>
</table>

Numbers from 2007
Introduction

Reasons for unexpected disruptions

- Infrastructure malfunctions
  - Rails, switches, catenary, bridges
- Computer problems in control centers
- Rolling stock breakdowns
- Accidents with other traffic
- Weather conditions
- Crew no shows
- ...

<table>
<thead>
<tr>
<th>Disruptions</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>933</td>
</tr>
<tr>
<td>Medium</td>
<td>1011</td>
</tr>
<tr>
<td>Large</td>
<td>834</td>
</tr>
</tbody>
</table>
Problem description

Given a blocked route and an estimated duration

- The timetable has been modified according to emergency scenarios
- The rolling stock has been rescheduled

Crew rescheduling

- Cover as much tasks as possible such that:
  - each original duty gets a feasible extension
  - the modifications to the crew schedule and the usage of taxis is as minimal as possible
Feasible extensions - example Gn 107

<table>
<thead>
<tr>
<th>Gn</th>
<th>ZI</th>
<th>Emn</th>
<th>ZI</th>
<th>Gn</th>
<th>On</th>
<th>Gn</th>
</tr>
</thead>
<tbody>
<tr>
<td>546</td>
<td>12:44</td>
<td>8045</td>
<td>15:28</td>
<td>8058</td>
<td>16:35</td>
<td>9157</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Feasible extensions - example Gn 107

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Feasible extensions - example Gn 107

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
**Notation**

- $N$: Set of tasks, where for every $i \in N$
  - $f_i$: Cost for canceling task $i$
- $\Delta$: Set of original duties
- $K^\delta$: Set of all feasible extensions for original duty $\delta \in \Delta$
  - $c_k^\delta$: Cost of extension $k$ for original duty $\delta$
- $x_k^\delta = \begin{cases} 
  1, & \text{if extension } k \text{ is selected for original duty } \delta \\
  0, & \text{otherwise} 
\end{cases}$
- $y_i = \begin{cases} 
  1, & \text{if task } i \text{ is canceled} \\
  0, & \text{otherwise} 
\end{cases}$
Mathematical model

\[
\begin{align*}
\min & \sum_{\delta \in \Delta} \sum_{k \in K_{\delta}} c_k^\delta x_k^\delta + \sum_{i \in N} f_i y_i \\
\text{s.t.} & \sum_{\delta \in \Delta} \sum_{k \in K_{\delta}} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in N \\
& \sum_{k \in K_{\delta}} x_k^\delta = 1 \quad \forall \delta \in \Delta \\
& x_k^\delta \in \{0, 1\} \quad \forall \delta \in \Delta, \forall k \in K_{\delta} \\
& y_i \in \{0, 1\} \quad \forall i \in N
\end{align*}
\]
Mathematical model

**Observations**
- Only a few original duties need to be rescheduled in order to obtain a good solution
- The number of feasible extension for each original duty might be huge

**Solution approach**
- Consider a core problem containing only a subset of the original duties and tasks
- Use a column generation based heuristic to solve the core problem
Overview over the algorithm

1. Define an initial core problem
2. Compute an initial solution using CG heuristic
3. Get list with uncovered tasks
4. List empty?
   - Yes: STOP
   - No:
     - Update list of uncovered tasks
     - Explore neighborhood using CG heuristic
     - Remove a task from the list and define a neighborhood
Overview over the algorithm

1. Define an initial core problem
2. Compute an initial solution using CG heuristic
3. Get list with uncovered tasks
4. List empty?
   - YES: STOP
   - NO:
     - Update list of uncovered tasks
     - Explore neighborhood using CG heuristic
     - Remove a task from the list and define a neighborhood
Selection of the initial core problem

- Define $N_p$ as the subset of tasks, where at least on of the following conditions holds:
  1. The task is canceled or modified (rerouted)
  2. The task is performed on the obstructed route and the departure time of task $i$ lies in the interval $[t_0, t_1 + p]$
  3. The task is part of the same train as one of the tasks selected in 1 and 2

- The subset of original duties is now defined as $\overline{\Delta} := \{\delta \in \Delta : \delta \text{ covers at least one task in } N_p\}$.

- The core problem is given by $\overline{\Delta}$ and $\overline{N}$ where $\overline{N}$ the set of tasks covered by a original duty in $\overline{\Delta}$. 

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Selection of the initial core problem

- Define $N_p$ as the subset of tasks, where at least one of the following conditions holds:
  1. The task is canceled or modified (rerouted)
  2. The task is performed on the obstructed route and the departure time of task $i$ lies in the interval $[t_0, t_1 + p]$
  3. The task is part of the same train as one of the tasks selected in 1 and 2

- The subset of original duties is now defined as $\Delta := \{\delta \in \Delta : \delta \text{ covers at least one task in } N_p\}$.

- The core problem is given by $\Delta$ and $\mathcal{N}$ where $\mathcal{N}$ the set of tasks covered by a original duty in $\Delta$. 

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Selection of the initial core problem

- Define $N_p$ as the subset of tasks, where at least one of the following conditions holds:
  1. The task is canceled or modified (rerouted)
  2. The task is performed on the obstructed route and the departure time of task $i$ lies in the interval $[t_0, t_1 + p]$
  3. The task is part of the same train as one of the tasks selected in 1 and 2

- The subset of original duties is now defined as $\Delta := \{\delta \in \Delta : \delta \text{ covers at least one task in } N_p\}$.

- The core problem is given by $\Delta$ and $\overline{N}$ where $\overline{N}$ the set of tasks covered by a original duty in $\Delta$. 
A column generation heuristic to solve a core problem

1. Define an initial core problem
2. Compute an initial solution using CG heuristic
3. Get list with uncovered tasks
4. List empty?
   - YES: STOP
   - NO:
     - Update list of uncovered tasks
     - Explore neighborhood using CG heuristic
     - Remove a task from the list and define a neighborhood
A column generation heuristic to solve a core problem

The RMP of the core problem in the \( n \)th column generation iteration reads.

\[
\min \sum_{\delta \in \Delta} \sum_{k \in K_n^\delta} c_k^\delta x_k^\delta + \sum_{i \in \overline{N}} f_i y_i
\]

\( s.t. \) \[
\sum_{\delta \in \Delta} \sum_{k \in K_n^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in \overline{N}
\]

\[
\sum_{k \in K_n^\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta
\]

\[
x_k^\delta \in \{0, 1\} \quad \forall \delta \in \Delta, \forall k \in K_n^\delta
\]

\[
y_i \in \{0, 1\} \quad \forall i \in \overline{N}
\]
Lagrangian relaxation

If we relax the set covering constraints, the Lagrangian subproblem becomes

\[ \phi(\lambda) = \min \sum_{\delta \in \Delta} \sum_{k \in K^\delta} c_k^\delta x_k^\delta + \sum_{i \in \bar{N}} f_i y_i + \sum_{i \in \bar{N}} \lambda_i (1 - \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a_{ikx}^\delta - y_i) \]

\[ \phi(\lambda) = \min \sum_{i \in \bar{N}} \lambda_i + \sum_{\delta \in \Delta} \sum_{k \in K_n^\delta} (c_k^\delta - \sum_{i \in \bar{N}} \lambda_i a_{ikx}^\delta) x_k^\delta + \sum_{i \in \bar{N}} (f_i - \lambda_i) y_i \]

s.t. \[ \sum_{k \in K_n^\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta \]

\[ x_k^\delta \in \{0, 1\} \quad \forall \delta \in \Delta, \forall k \in K_n^\delta \]

\[ y_i \in \{0, 1\} \quad \forall i \in \bar{N} \]
Lagrangian relaxation

If we relax the set covering constraints, the Lagrangian subproblem becomes

$$
\phi(\lambda) = \min \sum_{\delta \in \Delta} \sum_{k \in K_\delta} c_k^\delta x_k^\delta + \sum_{i \in \overline{N}} f_i y_i + \sum_{i \in \overline{N}} \lambda_i (1 - \sum_{\delta \in \Delta} \sum_{k \in K_\delta} a_{ik}^\delta x_k^\delta - y_i)
$$

$$
\phi(\lambda) = \min \sum_{i \in \overline{N}} \lambda_i + \sum_{\delta \in \Delta} \sum_{k \in K_\delta} (c_k^\delta - \sum_{i \in \overline{N}} \lambda_i a_{ik}^\delta) x_k^\delta + \sum_{i \in \overline{N}} (f_i - \lambda_i) y_i
$$

s.t.

$$
\sum_{k \in K_\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta
$$

$$
x_k^\delta \in \{0, 1\} \quad \forall \delta \in \overline{\Delta}, \forall k \in K_\delta
$$

$$
y_i \in \{0, 1\} \quad \forall i \in \overline{N}
$$

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Building blocks of the CG heuristic

Master problem

Apply Lagrangian relaxation and subgradient optimization

Pricing problems

Solve a resource constraint shortest path problem for each original duty on a dedicated acyclic graph

Feasible solutions

Use the Lagrangian multipliers in a greedy procedure to construct feasible solutions
Building blocks of the CG heuristic

**Master problem**

Apply Lagrangian relaxation and subgradient optimization

**Pricing problems**

Solve a resource constraint shortest path problem for each original duty on a dedicated acyclic graph

**Feasible solutions**

Use the Lagrangian multipliers in a greedy procedure to construct feasible solutions
Building blocks of the CG heuristic

**Master problem**
Apply Lagrangian relaxation and subgradient optimization

**Pricing problems**
Solve a resource constraint shortest path problem for each original duty on a dedicated acyclic graph

**Feasible solutions**
Use the Lagrangian multipliers in a greedy procedure to construct feasible solutions
Defining a neighborhood given an uncovered task

1. Define an initial core problem
2. Compute an initial solution using CG heuristic
3. Get list with uncovered tasks
4. List empty?
   - YES: STOP
   - NO: Explore neighborhood using CG heuristic
5. Remove a task from the list and define a neighborhood
6. Update list of uncovered tasks
Defining a neighborhood given an uncovered task - step 1

- Select the original duties, such that the driver has the route knowledge for task $j$, covering the $r$ closest departures before and after task $j$ from the same station.
- Select the original duty covering $\hat{j}$, the first task in reverse direction such that $\hat{j}$ can be performed directly after $j$. 

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Defining a neighborhood given an uncovered task - step 1

- Select the original duties, such that the driver has the route knowledge for task $j$, covering the $r$ closest departures before and after task $j$ from the same station.

- Select the original duty covering $\hat{j}$, the first task in reverse direction such that $\hat{j}$ can be performed directly after $j$. 

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Defining a neighborhood given an uncovered task - step 1

- Select the original duties, such that the driver has the route knowledge for task $j$, covering the $r$ closest departures before and after task $j$ from the same station.
- Select the original duty covering $\hat{j}$, the first task in reverse direction such that $\hat{j}$ can be performed directly after $j$. 
Defining a neighborhood given an uncovered task - step 2

- Choose for every original duty $\sigma$ selected in step 1 the $s$ most similar original duties

Where similarity is computed as

- a bonus for original duties from the same depot and
- the number of swap opportunities

A swap opportunity occurs if original duties $\sigma$ and $\tau$ cover tasks that depart from the same station around the same time
Defining a neighborhood given an uncovered task - step 2

Choose for every original duty $\sigma$ selected in step 1 the $s$ most similar original duties

Where similarity is computed as

- a bonus for original duties from the same depot and
- the number of swap opportunities

A swap opportunity occurs if original duties $\sigma$ and $\tau$ cover tasks that depart from the same station around the same time.
Defining a neighborhood given an uncovered task - step 2

- Choose for every original duty $\sigma$ selected in step 1 the $s$ most similar original duties

Where similarity is computed as

- a bonus for original duties from the same depot and
- the number of swap opportunities

- A swap opportunity occurs if original duties $\sigma$ and $\tau$ cover tasks that depart from the same station around the same time
Defining a neighborhood given an uncovered task - step 2

- Choose for every original duty $\sigma$ selected in step 1 the $s$ most similar original duties

Where similarity is computed as

- a bonus for original duties from the same depot and
- the number of swap opportunities

- A swap opportunity occurs if original duties $\sigma$ and $\tau$ cover tasks that depart from the same station around the same time
## Cost structure

<table>
<thead>
<tr>
<th>Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>Cost for canceling a task (A-B)</td>
</tr>
<tr>
<td>3000</td>
<td>Cost for canceling a round task (A-A)</td>
</tr>
<tr>
<td>400</td>
<td>Cost for changing an original duty</td>
</tr>
<tr>
<td>50</td>
<td>Cost for assigning a task to another original duty</td>
</tr>
<tr>
<td>1</td>
<td>Cost for using a transfer that was not in the original duty</td>
</tr>
<tr>
<td>1000</td>
<td>Cost for using a taxi</td>
</tr>
</tbody>
</table>
## Typical examples

Neighborhood $(r = 4, s = 3)$, no reserve duties

<table>
<thead>
<tr>
<th></th>
<th>It</th>
<th>LNS</th>
<th>LB</th>
<th>UB</th>
<th>A-B</th>
<th>A-A</th>
<th>SC</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ztm B</td>
<td>1</td>
<td>38096</td>
<td>35555</td>
<td>38096</td>
<td>1</td>
<td>1</td>
<td>423</td>
<td>47</td>
</tr>
<tr>
<td>Ztm B</td>
<td>2</td>
<td>16303</td>
<td>3217</td>
<td>3217</td>
<td>0</td>
<td>0</td>
<td>200</td>
<td>53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>It</th>
<th>LNS</th>
<th>LB</th>
<th>UB</th>
<th>A-B</th>
<th>A-A</th>
<th>SC</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ht B</td>
<td>1</td>
<td>129004</td>
<td>62894</td>
<td>129004</td>
<td>4</td>
<td>3</td>
<td>658</td>
<td>206</td>
</tr>
<tr>
<td>Ht B</td>
<td>2</td>
<td>77332</td>
<td>43343</td>
<td>46618</td>
<td>1</td>
<td>4</td>
<td>415</td>
<td>310</td>
</tr>
<tr>
<td>Ht B</td>
<td>3</td>
<td>74684</td>
<td>34888</td>
<td>34888</td>
<td>1</td>
<td>3</td>
<td>352</td>
<td>321</td>
</tr>
<tr>
<td>Ht B</td>
<td>4</td>
<td>72894</td>
<td>30675</td>
<td>32236</td>
<td>1</td>
<td>2</td>
<td>381</td>
<td>360</td>
</tr>
<tr>
<td>Ht B</td>
<td>5</td>
<td>72792</td>
<td>27658</td>
<td>27659</td>
<td>1</td>
<td>2</td>
<td>338</td>
<td>380</td>
</tr>
<tr>
<td>Ht B</td>
<td>6</td>
<td>71196</td>
<td>28920</td>
<td>28920</td>
<td>1</td>
<td>1</td>
<td>306</td>
<td>402</td>
</tr>
</tbody>
</table>
## Typical examples

Neighborhood \((r = 4, s = 3)\), no reserve duties

<table>
<thead>
<tr>
<th>It</th>
<th>LNS</th>
<th>LB</th>
<th>UB</th>
<th>A-B</th>
<th>A-A</th>
<th>SC</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ztm B</td>
<td>1</td>
<td>38096</td>
<td>35555</td>
<td>38096</td>
<td>1</td>
<td>1</td>
<td>423</td>
</tr>
<tr>
<td>Ztm B</td>
<td>2</td>
<td>16303</td>
<td>3217</td>
<td>3217</td>
<td>0</td>
<td>0</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>It</th>
<th>LNS</th>
<th>LB</th>
<th>UB</th>
<th>A-B</th>
<th>A-A</th>
<th>SC</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ht B</td>
<td>1</td>
<td>129004</td>
<td>62894</td>
<td>129004</td>
<td>4</td>
<td>3</td>
<td>658</td>
</tr>
<tr>
<td>Ht B</td>
<td>2</td>
<td>77332</td>
<td>43343</td>
<td>46618</td>
<td>1</td>
<td>4</td>
<td>415</td>
</tr>
<tr>
<td>Ht B</td>
<td>3</td>
<td>74684</td>
<td>34888</td>
<td>34888</td>
<td>1</td>
<td>3</td>
<td>352</td>
</tr>
<tr>
<td>Ht B</td>
<td>4</td>
<td>72894</td>
<td>30675</td>
<td>32236</td>
<td>1</td>
<td>2</td>
<td>381</td>
</tr>
<tr>
<td>Ht B</td>
<td>5</td>
<td>72792</td>
<td>27658</td>
<td>27659</td>
<td>1</td>
<td>2</td>
<td>338</td>
</tr>
<tr>
<td>Ht B</td>
<td>6</td>
<td>71196</td>
<td>28920</td>
<td>28920</td>
<td>1</td>
<td>1</td>
<td>306</td>
</tr>
</tbody>
</table>
Summary of results from 10 instances

- For 3 instances all tasks are covered in the solution found for the initial core problem
- For 7 instances some tasks need to be canceled in the solution found for the initial core problem
  - For 2 of these instances the final solution is better than the lower bound for the initial core problem
  - For 3 of these instances no tasks need to be canceled in the final solution

D. Potthoff
A Column Generation based Heuristic for Train Driver Rescheduling
Summary of results from 10 instances

- For 3 instances all tasks are covered in the solution found for the initial core problem.
- For 7 instances some tasks need to be canceled in the solution found for the initial core problem:
  - For 2 of these instances the final solution is better than the lower bound for the initial core problem.
  - For 3 of these instances no tasks need to be canceled in the final solution.
Summary of results from 10 instances

- For 3 instances all tasks are covered in the solution found for the initial core problem.
- For 7 instances some tasks need to be canceled in the solution found for the initial core problem.
  - For 2 of these instances the final solution is better than the lower bound for the initial core problem.
  - For 3 of these instances no tasks need to be canceled in the final solution.
Summary of results from 10 instances

- For 3 instances all tasks are covered in the solution found for the initial core problem
- For 7 instances some tasks need to be canceled in the solution found for the initial core problem
  - For 2 of these instances the final solution is better than the lower bound for the initial core problem
  - For 3 of these instances no tasks need to be canceled in the final solution
Conclusion

Observations

- Good solutions can be obtained within a couple of minutes for real world instances
- Some initial solutions can be improved by exploring the neighborhood of uncovered tasks

Now we are going to ...

- Improve the solutions for the initial core problems
  - Apply column fixing
  - Refine the greedy procedure
- Develop additional neighborhood definitions and integrate them in the overall scheme
Thank you for your attention!