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
- ...

Numbers from 2007

Disruptions	#
Small	933
Medium	1011
Large	834

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
- ...

Numbers from 2007

Disruptions	#
Small	933
Medium	1011
Large	834

Problem description

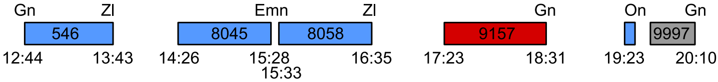
Given a blocked route and an estimated duration

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

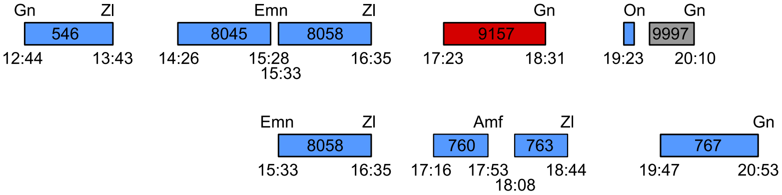
Crew rescheduling

- 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

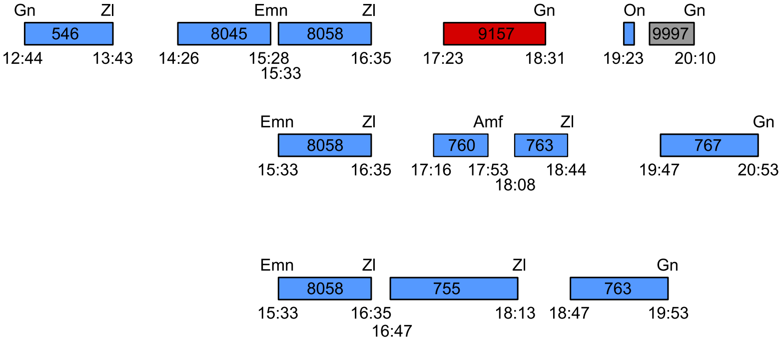
Feasible extensions - example Gn 107



Feasible extensions - example Gn 107



Feasible extensions - example Gn 107



Notation

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

Mathematical model

$$\min \sum_{\delta \in \Delta} \sum_{k \in K^\delta} c_k^\delta x_k^\delta + \sum_{i \in N} f_i y_i \quad (1)$$

$$\text{s.t.} \quad \sum_{\delta \in \Delta} \sum_{k \in K^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in N \quad (2)$$

$$\sum_{k \in K^\delta} x_k^\delta = 1 \quad \forall \delta \in \Delta \quad (3)$$

$$x_k^\delta \in \{0,1\} \quad \forall \delta \in \Delta, \forall k \in K^\delta \quad (4)$$

$$y_i \in \{0,1\} \quad \forall i \in N \quad (5)$$

Mathematical model

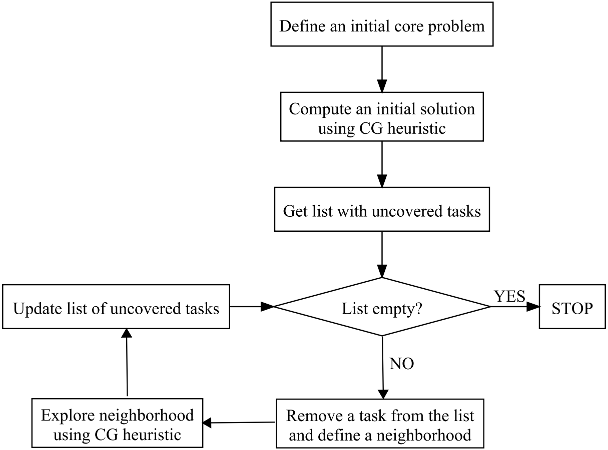
Observations

- 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

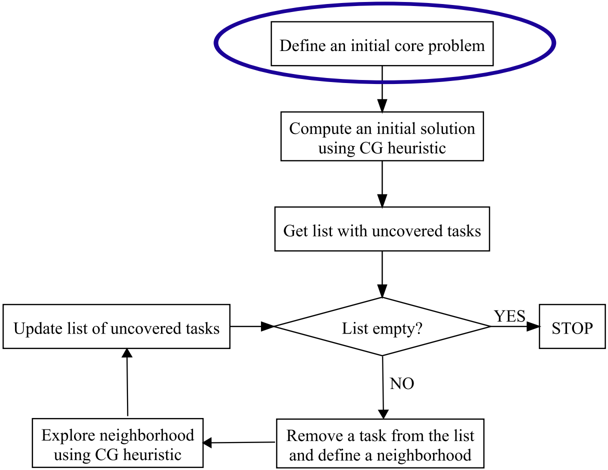
Solution approach

- 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



Overview over the algorithm



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 $\overline{\Delta} := \{\delta \in \Delta : \delta$ 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 $\overline{\Delta}$.

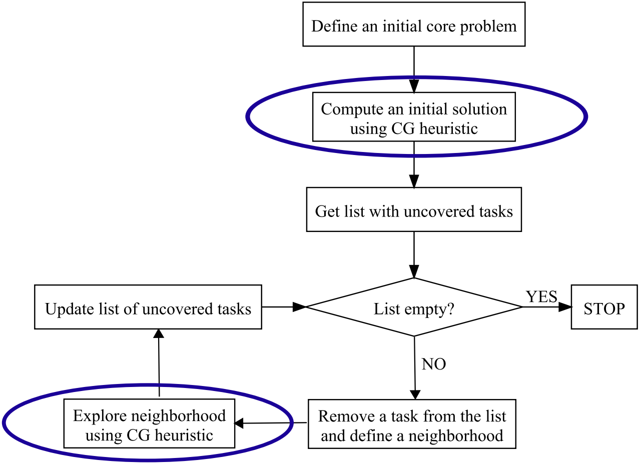
Selection of the initial core problem

- 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 $\overline{\Delta} := \{\delta \in \Delta : \delta$ 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 $\overline{\Delta}$.

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 $\overline{\Delta} := \{\delta \in \Delta : \delta$ 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 $\overline{\Delta}$.

A column generation heuristic to solve a core problem



A column generation heuristic to solve a core problem

The RMP of the core problem in the *n*th column generation iteration reads.

$$\min \sum_{\delta \in \overline{\Delta}} \sum_{k \in K_n^\delta} c_k^\delta x_k^\delta \quad + \quad \sum_{i \in \overline{N}} f_i y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{\delta \in \overline{\Delta}} \sum_{k \in K_n^\delta} a_{ik}^\delta x_k^\delta + y_i \geq 1 \quad \forall i \in \overline{N} \tag{2}$$

$$\sum_{k \in K_n^\delta} x_k^\delta = 1 \quad \forall \delta \in \overline{\Delta} \tag{3}$$

$$x_k^\delta \in \{0, 1\} \quad \forall \delta \in \overline{\Delta}, \forall k \in K_n^\delta \tag{4}$$

$$y_i \in \{0, 1\} \quad \forall i \in \overline{N} \tag{5}$$

Lagrangian relaxation

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

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

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

$$\text{s.t.} \quad \sum_{k \in K_n^\delta} x_k^\delta = 1 \quad \forall \delta \in \overline{\Delta}$$

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

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

Lagrangian relaxation

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

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

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

$$\text{s.t.} \quad \sum_{k \in K_n^\delta} x_k^\delta = 1 \quad \forall \delta \in \overline{\Delta}$$

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

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

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

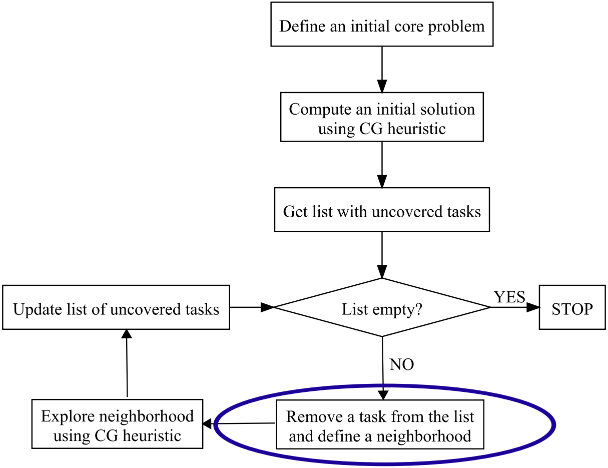
Pricing problems

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

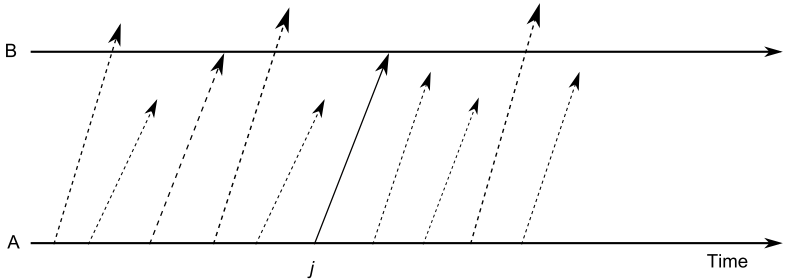
Feasible solutions

Use the Lagrangian multipliers in a greedy procedure to construct feasible solutions

Defining a neighborhood given an uncovered task

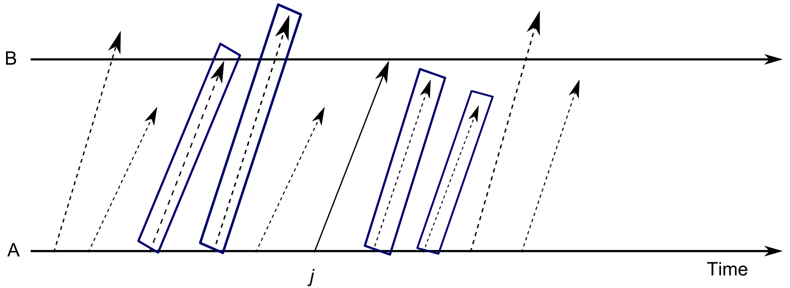


Defining a neighborhood given an uncovered task - step1



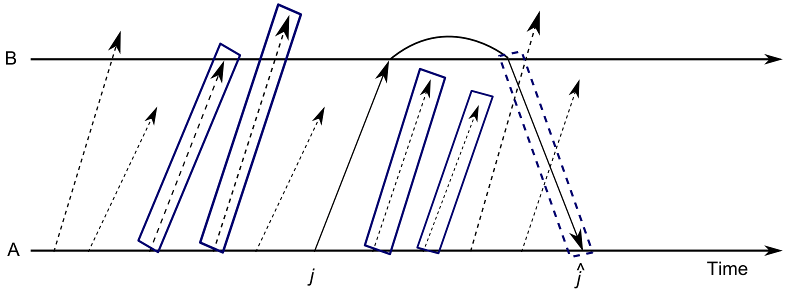
- Select the original duties, such that the driver has the route knowledge for task j, covering the r closest departures before and after task j from the same station
- Select the original duty covering $\hat{j}$, the first task in reverse direction such that $\hat{j}$ can be performed directly after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

Defining a neighborhood given an uncovered task - step1



- 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 - step1



- 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 σ 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 σ and τ 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 σ 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 σ and τ 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 σ 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 σ and τ 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 σ 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 σ and τ cover tasks that depart from the same station around the same time

Cost structure

20000	Cost for canceling a task (A-B)
3000	Cost for canceling a round task (A-A)
400	Cost for changing an original duty
50	Cost for assigning a task to another original duty
1	Cost for using a transfer that was not in the original duty
1000	Cost for using a taxi

Typical examples

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

	It	LNS	LB	UB	A-B	A-A	SC	Time
Ztm B	1	38096	35555	38096	1	1	423	47
Ztm B	2	16303	3217	3217	0	0	200	53

	It	LNS	LB	UB	A-B	A-A	SC	Time
Ht B	1	129004	62894	129004	4	3	658	206
Ht B	2	77332	43343	46618	1	4	415	310
Ht B	3	74684	34888	34888	1	3	352	321
Ht B	4	72894	30675	32236	1	2	381	360
Ht B	5	72792	27658	27659	1	2	338	380
Ht B	6	71196	28920	28920	1	1	306	402

Typical examples

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

	It	LNS	LB	UB	A-B	A-A	SC	Time
Ztm B	1	38096	35555	38096	1	1	423	47
Ztm B	2	16303	3217	3217	0	0	200	53

	It	LNS	LB	UB	A-B	A-A	SC	Time
Ht B	1	129004	62894	129004	4	3	658	206
Ht B	2	77332	43343	46618	1	4	415	310
Ht B	3	74684	34888	34888	1	3	352	321
Ht B	4	72894	30675	32236	1	2	381	360
Ht B	5	72792	27658	27659	1	2	338	380
Ht B	6	71196	28920	28920	1	1	306	402

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

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 be 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!