The Prominence of Stabilization Techniques in Column Generation: the case of Freight Transportation

Ruslan Sadykov\textsuperscript{1,2} \hspace{1cm} Alexander A. Lazarev\textsuperscript{3}
Arthur Pessoa\textsuperscript{4} \hspace{1cm} Eduardo Uchoa\textsuperscript{4}
François Vanderbeck\textsuperscript{2,1}

\textsuperscript{1} Inria Bordeaux, Talence, France \hspace{1cm} \textsuperscript{2} University Bordeaux, Talence, France
\textsuperscript{3} Institute of Control Sciences, Moscow, Russia \hspace{1cm} \textsuperscript{4} Univ. Federal Fluminense, Niteroi, Brazil

Odysseus 2015
Ajaccio, France, June 1
Contents

Freight railcar routing application

Column generation approach

Stabilization

Results and conclusions
The freight car routing application: overview

initial car distribution

transportation demands
Specificity of freight rail transportation in Russia

The state company

- Freight car blocking
- Freight train scheduling
- Locomotives management
- Personnel management

Independent freight car management companies

- Assignment of transportation demands to freight cars
- Freight car routing

Transp. costs matrix
Transp. times matrix

car movements

Distances are large, and average freight train speed is low (≈ 300 km/day): discretization in periods of 1 day is reasonable
Specificity of freight rail transportation in Russia

The state company

- Freight car blocking
- Freight train scheduling
- Locomotives management
- Personnel management

Independent freight car management companies

- Assignment of transportation demands to freight cars
- Freight car routing

Distances are large, and average freight train speed is low ($\approx 300 \text{ km/day}$): discretization in periods of 1 day is reasonable.
The freight car routing application: input and output

Input

▶ Railroad network (stations)
▶ Initial location of cars (sources)
▶ Transportation demands and associated profits
▶ Transportation times between stations
▶ **Costs**: transfer costs and standing (waiting) daily rates;

Output

▶ A set of accepted demands and their execution dates
▶ Empty and loaded cars movements to meet the demands (car routing)

Objective

Maximize the total net profit
The freight car routing application: input and output

Input

▶ Railroad network (stations)
▶ Initial location of cars (sources)
▶ Transportation demands and associated profits
▶ Transportation times between stations
▶ Costs: transfer costs and standing (waiting) daily rates;

Output: operational plan

▶ A set of accepted demands and their execution dates
▶ Empty and loaded cars movements to meet the demands (car routing)

Objective

Maximize the total net profit
Similar applications in the literature

[Fukasawa et al., 2002]

- Train schedule is known
- Cars should be assigned to trains to be transported
- Discretization by the moments of arrival and departure of trains.
- Smaller time horizon (7 days)
Similar applications in the literature

[Fukasawa et al., 2002]
- Train schedule is known
- Cars should be assigned to trains to be transported
- Discretization by the moments of arrival and departure of trains.
- Smaller time horizon (7 days)

Other works
- [Holmberg et al., 1998]
- [Löbel, 1998]
- [Lulli et al., 2011]
- [Caprara et al., 2011]
Commodity graph [Stratennikov and Shiryaev, 2012]

Commodity \( c \in C \) represents the flow (movements) of cars of type \( c \).

Graph \( G_c = (V_c, A_c) \) for commodity \( c \in C \):

Each vertex \( v \in V_c \) represents the location of cars of type \( c \) on a certain station at a certain time, standing at a certain rate \( g_a \) — cost of arc \( a \in A_c \).
Multi-commodity flow formulation

Notations

$Q$ — set of demands,
$C_q$ — set of car types compatible with demand $q \in Q$,
$n_{\text{max}}^q$ — maximum number of cars to assign to demand $q \in Q$,
$\vec{n}_v^c$ — number of cars of type $c$ situated initially in vertex $v \in V$.

Variables

$x_{c,a}^c \in \mathbb{Z}_+$ — flow size along arc $a \in A_c$, $c \in C$
Multi-commodity flow formulation

Notations

- $Q$ — set of demands,
- $C_q$ — set of car types compatible with demand $q \in Q$,
- $n_{q}^{\text{max}}$ — maximum number of cars to assign to demand $q \in Q$,
- $\vec{n}_v^c$ — number of cars of type $c$ situated initially in vertex $v \in V$.

Variables

- $x_{a}^{c} \in \mathbb{Z}_{+}$ — flow size along arc $a \in A_c$, $c \in C$

\[
\begin{align*}
\min & \quad \sum_{c \in C} \sum_{a \in A_c} g_a x_{a}^{c} \\
\sum_{c \in C_q} \sum_{a \in A_{cq}} x_{a}^{c} & \leq n_{q}^{\text{max}} \quad \forall q \in Q \\
\sum_{a \in \delta^-(v)} x_{a}^{c} - \sum_{a \in \delta^+(v)} x_{a}^{c} & = \vec{n}_v^c \quad \forall c \in C, v \in V_c \\
x_{a}^{c} & \in \mathbb{Z}_{+} \quad \forall c \in C, a \in V_c
\end{align*}
\]
Contents

Freight railcar routing application

Column generation approach

Stabilization

Results and conclusions
Path reformulation

- $S_c$ — set of type $c$ car “sources” = $\{v \in V_c : \bar{n}_v > 0\}$
- $P^c_s$ — set of paths (type $c$ car routes) from source $s \in S_c$

Variables

- $\lambda_p$ — flow size along path $p \in P^c_s$, $s \in S_c$, $c \in C$

\[
\begin{align*}
\min \, & \sum_{c \in C} \sum_{s \in S_c} \sum_{p \in P^c_s} g^\text{path}_p \lambda_p \\
\sum_{c \in C} \sum_{s \in S_c} \sum_{p \in P^c_s : q \in Q^\text{path}_p} \lambda_p & \leq n^\text{max}_q \quad \forall q \in Q \\
\sum_{p \in P^c_s} \lambda_p & = \bar{n}^c_s \quad \forall c \in C, s \in S_c \\
\lambda_p & \in \mathbb{Z}_+ \quad \forall c \in C, s \in S_c, p \in P^c_s
\end{align*}
\]
Column generation for path reformulation

- Pricing problem decomposes to shortest path problems, one for each source
  - slow: number of sources are thousands

To accelerate, for each $c \in C$, we search for an in-tree of shortest paths to the terminal vertex from all $s \in S_c$

- drawback: some demands are severely “overcovered”, bad convergence

We developed an iterative procedure:

repeat

Find an in-tree $T$ from all non-exhausted sources;

foreach path $p$ in $T$ in the order of its reduced cost do

- Find $n' –$ the maximum number of cars able to follow $p$;
- if $n' > 0$ then
  - Add $\lambda_p$ to the restricted master;
  - Reduce by $n'$ the number of cars in the source of $p$;
  - Reduce by $n'$ the volume of all demands covered by $p$;

until iteration limit $k$ or all demands are covered or all sources are exhausted;
Column generation for path reformulation

- Pricing problem decomposes to shortest path problems, one for each source
  - **slow**: number of sources are thousands
- To accelerate, for each \( c \in C \), we search for an in-tree of shortest paths to the terminal vertex from all \( s \in S_c \)
  - **drawback**: some demands are severely “overcovered”, bad convergence
Column generation for path reformulation

- Pricing problem decomposes to shortest path problems, one for each source
  - slow: number of sources are thousands
- To accelerate, for each \( c \in C \), we search for an in-tree of shortest paths to the terminal vertex from all \( s \in S_c \)
  - drawback: some demands are severely “overcovered”, bad convergence
- We developed an iterative procedure:

\[
\text{repeat}\ \\
\quad \text{Find an in-tree } T \text{ from all non-exhausted sources;} \\
\quad \text{foreach path } p \text{ in } T \text{ in the order of its reduced cost do}\ \\
\quad \quad \text{Find } n' - \text{ the maximum number of cars able to follow } p; \\
\quad \quad \text{if } n' > 0 \text{ then}\ \\
\quad \quad \quad \text{Add } \lambda_p \text{ to the restricted master;} \\
\quad \quad \quad \text{Reduce by } n' \text{ the number of cars in the source of } p; \\
\quad \quad \quad \text{Reduce by } n' \text{ the volume of all demands covered by } p; \\
\quad \text{until iteration limit } k \text{ or all demands are covered or all sources are exhausted;}
\]
Diving Heuristic

Master problem solution $\lambda^*$ can be fractional, so we apply the diving heuristic [Joncour et al., 2010]

- use Depth-First Search
- at each node of the tree
  - select \textit{least fractional} column $\bar{\lambda}_p$:
    - rounded value $\lceil \bar{\lambda}_p \rceil > 0$
  - add $\lceil \bar{\lambda}_p \rceil$ to the partial solution
  - update right-hand-side of the master constraints
  - apply \textit{preprocessing}, which may lead to a change in the pricing problem variables bounds
  - solve the updated master LP
Contents

Freight railcar routing application

Column generation approach

Stabilization

Results and conclusions
Column generation in the dual space

$L(\pi)$ — Lagrangian dual function

Outer approximation
Column generation in the dual space

$L(\pi) —$ Lagrangian dual function

Outer approximation
Column generation in the dual space

$L(\pi) —$ Lagrangian dual function

Outer approximation

Inner approximation
Dual Price Smoothing [Wentges, 1997]

\[ \tilde{\pi}^t = \alpha \hat{\pi} + (1 - \alpha) \pi^t \]

\((\pi^{\text{in}}, \eta^{\text{in}}) := (\hat{\pi}, \hat{\ell})\)

\((\pi^{\text{out}}, \eta^{\text{out}}) := (\pi^t, \eta^t)\)

\((\pi^{\text{sep}}, \eta^{\text{sep}}) := \alpha (\pi^{\text{in}}, \eta^{\text{in}}) + (1 - \alpha) (\pi^{\text{out}}, \eta^{\text{out}})\)
Auto-adaptative $\alpha$-schedule [Pessoa et al., 2014]

$L(\pi) = \hat{L}$

$\pi^{\text{out}}$

$\pi^{\text{in}}$

$\pi$

$g(\tilde{\pi})$

increase $\alpha$

decrease $\alpha$
Penalty functions [du Merle et al., 1999]

\[ \pi^t = \arg\max_{\pi \in \mathbb{R}_+^m} \left\{ L^t(\pi) - \hat{S}_t(\pi) \right\} \]

Here:

\[ \Delta = \frac{\sum_{q \in Q} |\pi_q^1 - \hat{\pi}_q^0|}{|Q| \cdot \kappa} \]
Contents

Freight railcar routing application

Column generation approach

Stabilization

Results and conclusions
Stabilization results

**Real-life instances**: 40-140 days horizon, 1,025 stations, up to 5,300 demands, 11 car types, 12,651 cars, and 8,232 sources. Up to \( \approx 230 \) thousands nodes and 7.5 millions arcs.
Stabilization results

**Real-life instances**: 40-140 days horizon, 1,025 stations, up to 5,300 demands, 11 car types, 12,651 cars, and 8,232 sources. Up to \( \approx 230 \) thousands nodes and 7.5 millions arcs.

<table>
<thead>
<tr>
<th>size</th>
<th>CG</th>
<th>Smooth</th>
<th>Pen</th>
<th>Comb</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>23</td>
<td>19</td>
<td>32</td>
<td>21</td>
</tr>
<tr>
<td>60</td>
<td>117</td>
<td>73</td>
<td>155</td>
<td>95</td>
</tr>
<tr>
<td>80</td>
<td>570</td>
<td>234</td>
<td>178</td>
<td>114</td>
</tr>
<tr>
<td>100</td>
<td>2481</td>
<td>607</td>
<td>278</td>
<td>152</td>
</tr>
<tr>
<td>120</td>
<td>8947</td>
<td>1465</td>
<td>410</td>
<td>213</td>
</tr>
<tr>
<td>140</td>
<td>28884</td>
<td>3069</td>
<td>756</td>
<td>338</td>
</tr>
</tbody>
</table>
Diving heuristic results

All instances solved to optimality by the diving heuristic. (means Lagrangian bound is equal to the optimal solution value).
Diving heuristic results

All instances solved to optimality by the diving heuristic.
(means Lagrangian bound is equal to the optimal solution value).

![Graph showing solution time against planning horizon length for different methods including Column generation, Column generation + diving, and Cplex 12.4.]

<table>
<thead>
<tr>
<th>size</th>
<th>CG</th>
<th>+Diving</th>
<th>Cplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>21</td>
<td>22</td>
<td>51</td>
</tr>
<tr>
<td>60</td>
<td>95</td>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>80</td>
<td>114</td>
<td>145</td>
<td>245</td>
</tr>
<tr>
<td>100</td>
<td>152</td>
<td>211</td>
<td>408</td>
</tr>
<tr>
<td>120</td>
<td>213</td>
<td>344</td>
<td>633</td>
</tr>
<tr>
<td>140</td>
<td>338</td>
<td>377</td>
<td>1127</td>
</tr>
</tbody>
</table>
Conclusions

- A freight car routing application can be modelled as the multi-commodity flow problem.
- Non-stabilized column generation implementation is not competitive with Cplex.
- Generic combined stabilization with a single parameter gives up to 85x speed-up.
- Generic diving heuristic allows us to obtain optimal integer solutions for real-life instances up to 3 times faster than Cplex.
- Column-and-row generation [Sadykov et al., 2013] gives better results than the stabilized column generation, but the diving heuristic cannot be directly applied.


