Swiss Freight Railway Network Design Problem

Nikola Obrenović, Virginie Lurkin, Stefano Bortolomoiol, Michel Bierlaire
Transport and Mobility Laboratory TRANSP-OR
École Polytechnique Fédérale de Lausanne EPFL

Vincent J. Baeriswyl, Jasmin Bigdon
SBB Cargo AG, Olten, Switzerland

hEART 2018, Athens, Greece
Outline

1. Problem introduction
2. Problem definition
3. Heuristic algorithm
4. Algorithm results
5. Conclusions and future work
Marshaling and shunting yards

- Bundling different commodities with close origins and close destinations

Diagram:

- Marshaling yard
- Shunting yard
- Station
Problem setting

- Existing SBB Cargo network
  - 2 inner marshaling yards
  - 3 border marshaling yards
  - Approx. 70 shunting yards
    – 50 can be changed

- Solution should provide:
  - Optimal number and locations of marshaling and shunting yards
  - Set of used trains
  - Assignment of commodities to trains
Problem definition

- Combination and extension of the HLP and MNDP
- Network elements:
  - $N$ – Set of stations, including potential marshaling and shunting yards
  - $A$ – Set of direct links between the stations
  - $K$ – Set of transported commodities each described with the origin, destination, weight and number of wagons
- Objective function:

$$\min C_{ct} + C_{ls} + C_{cs}$$

Commodity transport costs

Locomotive and staff costs

Commodity shunting costs
Problem definition (cont.)

• Commodity transport costs:

\[ C_{ct} = \sum_{k \in K} \sum_{p \in N} \sum_{q \in N} \sum_{(i,j) \in A} d_{ij} w^k P_{w} x_{ij}^{pq} f_{pq}^k \]

• Constants:
  • \( d_{ij} \) – Distance between nodes \( i \) and \( j \)
  • \( w^k \) – Weight of commodity \( k \)
  • \( P_{w} \) – Transport price per weight and distance unit

• Variables:
  • \( x_{ij}^{pq} \) - Determines if arc \((i,j)\) is used by the train between \( p \) and \( q \)
  • \( f_{pq}^k \) - Determines if commodity \( k \) is transported on the train between \( p \) and \( q \)
Problem definition (cont.)

- Locomotive and staff costs:
  \[
  C_{ls} = \sum_{p \in N} \sum_{q \in N} \sum_{(i,j) \in A} n_{pq} x_{ij}^{pq} d_{ij} P_L
  \]

- Constants:
  - \( d_{ij} \) – Distance between nodes \( i \) and \( j \)
  - \( P_L \) – Locomotive and staff cost per distance

- Variables:
  - \( x_{ij}^{pq} \) - Determines if arc \((i, j)\) is used by the train between \( p \) and \( q \)
  - \( n_{pq} \) - Number of trains between \( p \) and \( q \)
Problem definition (cont.)

- Commodity shunting costs:

\[ C_{cs} = \sum_{k \in K} \sum_{i \in N} S v^k s_i^k + \sum_{k \in K} \sum_{i \in N} M v^k m_i^k \]

- Constants:
  - \( v^k \) – Number of wagons of commodity \( k \)
  - \( S \) – Shunting price per wagon, in a shunting yard
  - \( M \) – Shunting price per wagon, in a marshaling yard

- Variables:
  - \( s_i^k \) - Determines if commodity \( k \) is shunted in the shunting yard \( i \)
  - \( m_i^k \) - Determines if commodity \( k \) is shunted in the marshaling yard \( i \)
Problem definition (cont.)

- Constraints from MNDP:
  - Flow conservation constraints for trains
  - Arc capacity constrains
- Constraints from HLP:
  - Hub capacity constraints
  - Maximal number of hubs
- Node type constraints:
  \[
  r_i + s_i + m_i = 1, \quad \forall i \in N
  \]
  \[
  \sum_{k \in K} s_i^k \leq s_i M_1, \quad \forall i \in N
  \]
  \[
  \sum_{k \in K} m_i^k \leq m_i M_2, \quad \forall i \in N
  \]
  - Variables:
    - \( r_i \) - If node \( i \) is a regular station
    - \( s_i \) - If node \( i \) is a shunting yard
    - \( m_i \) - If node \( i \) is a marshaling yard
Problem definition (cont.)

- Commodity assignment constraints:
  \[ f_{pq}^k \leq s_p^k + m_p^k + o_{kp}, \quad \forall p, q \in N, \forall k \in K \]
  \[ f_{pq}^k \leq s_q^k + m_q^k + d_{kq}, \quad \forall p, q \in N, \forall k \in K \]

- Flow conservation constraints for commodities:
  \[ \sum_{q \in N} f_{pq}^k - \sum_{q \in N} f_{qp}^k = o_{kp} - d_{kp}, \quad \forall p \in N, \forall k \in K \]

- Constants:
  - \( o_{kp} \) – Determines if node \( p \) is the origin of commodity \( k \)
  - \( d_{kp} \) – Determines if node \( p \) is the destination of commodity \( k \)
• Train capacity constraints:

\[ \sum_{k \in K} f_{pq} \nu^k l^k \leq L_t n_{pq}, \quad \forall p, q \in N \]

• Constants:
  • \( l^k \) – Length of commodity \( k \)
  • \( L_t \) – Max. allowed train length
Input data

• Size of the SBB Cargo network:
  • Approx. 2100 stations
  • Approx. 2500 direct links
• Over 65000 commodities
  • Yearly demand, scaled to daily average
Heuristic algorithm

- Heuristic algorithm composed of 4 stages:
  - Yard location and sizing
  - Initial train generation
  - Commodity assignment (routing)
  - Train number reduction
Heuristic algorithm – Yard location and sizing

- Yard location:
  - Adaptive large neighborhood search
  - Variable neighborhood search
Heuristic algorithm - Neighborhoods

- Select the busiest station close to the MY
- Select the least used MY
- Distance-dependent probability of station selection
- Select fully utilized SY, with maximum capacity
Heuristic algorithm - Neighborhoods

- Select SY with most unused capacity
- Select fully utilized SY with below maximum capacity
- Select underused SY with minimum capacity
- Select frequently used regular station
Heuristic algorithm – Initial trains generation

- Marshaling yard
- Shunting yard
- Station
Heuristic algorithm - Path alternatives

- Via marshaling and shunting yards
  - Most often case
  - If the same marshaling yards is closest to both shunting yards
  - Skipped shunting yard
Heuristic algorithm - Path alternatives

- Direct (shortest) path
  - For large commodities

- Via shunting yards
  - For local transport
Heuristic algorithm – Commodity assignment

- Commodity routing:
  - Prioritized assignment algorithm

  Select commodity by priority

  Path alternatives test
  Via marshaling and shunting yards, shortest path, etc.

  Path selection
  Cheapest, available path

  Add necessary trains
  If none of the paths is feasible
Heuristic algorithm – Reduction of train number

- Remove all unused trains
- VNS loop:
  - Marshaling yard
  - Shunting yard
  - Station
Heuristic algorithm – Reduction of train number

• Remove all unused trains

• VNS loop:
  - Marshaling yard
  - Shunting yard
  - Station
Heuristic algorithm – Reduction of train number

• Remove all unused trains

• VNS loop:

- Marshaling yard
- Shunting yard
- Station

SBB CFF FFS Cargo
Heuristic algorithm – Reduction of train number

- Remove all unused trains
- VNS loop:
  - Marshaling yard
  - Shunting yard
  - Station
Algorithm results

- Best resulting networks with (S1) and without (S2) allowing increase in the number of marshaling yards:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>VNS</td>
<td>2h</td>
<td>10.01%</td>
</tr>
<tr>
<td>S2</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>VNS</td>
<td>4.5h</td>
<td>4.48%</td>
</tr>
</tbody>
</table>

- Daily transportation cost in the original network: over 2.5 Million CHF
- Business decision: S2
Algorithm results – Marshaling yards
Algorithm results – Shunting yards
Results analysis

- Costs of transportation are dominant over yard operation costs
- Costs of yard opening and maintenance are not taken into account
  - Would reduce the number of yards and their size
  - Opening new yards will be less favored by the algorithm
  - Could be included in another case study
- New yards can be near the existing ones
  - The objective function has been extended to penalize this situation
Results analysis - Routing
Conclusions

• Developed algorithm explores various network changes, their combinations and their influence to the transportation costs
  • Flexible, easily extendable algorithm
• The algorithm identified network changes resulting in transportation cost reduction
• The objective function should be extended with the real costs of maintenance of the marshaling and shunting yards
  • Relevant change in the algorithm result
Future work

- Algorithm parallelization
- Solve the problem exactly (on the subset of input data)
  - To benchmark the heuristic result
Exact solution approach

Train transshipment problem
- Trains flow

Flow decomposition algorithm
- Set of running trains

Commodities assignment
Exact solution approach (cont.)

- Open questions:
  - If the same node is both origin and destination for different commodities, the transshipment problem is aware only of the difference
  - No guaranties that the O-D demand will be satisfied
  - Yard location is missing
Thank you!

Questions?

nikola.obrenovic@epfl.ch
References


Related problems

- Hub location problem (HLP)
  - Missing track capacities and hub operation costs
- Multicommodity flow problem (MFP)
  - Missing hub capacities and operation costs
- Multicommodity network design problem (MNDP)
  - Missing hub types and operation costs
Problem definition (cont.)

- Flow conservation and routing constraints for trains:

\[
\sum_{(i,j) \in A} x_{ij}^{pq} - \sum_{(j,i) \in A} x_{ji}^{pq} = 0, \quad \forall p, q \in N, \forall i \in N \setminus \{p, q\}
\]

\[
\sum_{(p,i) \in A} x_{pi}^{pq} = t_{pq}, \quad \forall p, q \in N
\]

\[
\sum_{(i,q) \in A} x_{iq}^{pq} = t_{pq}, \quad \forall p, q \in N
\]

\[
\sum_{(i,j) \in A} x_{ij}^{pq} \leq t_{pq}, \quad \forall p, q \in N, \forall i \in N \setminus \{p, q\}
\]

\[
\sum_{(j,i) \in A} x_{ji}^{pq} \leq t_{pq}, \quad \forall p, q \in N, \forall i \in N \setminus \{p, q\}
\]

- Variables:
  - \( t_{pq} \) - Binary variable determining the existence of trains between \( p \) and \( q \)
Problem definition (cont.)

- Arc capacity constraints:
  \[ t_{pq} \leq n_{pq}, \quad \forall p, q \in N \]
  \[ n_{pq} \leq M t_{pq}, \quad \forall p, q \in N \]

\[ \sum_{p \in N} \sum_{q \in N} n_{pq} x_{ij}^{pq} \leq u_{ij}, \quad \forall (i, j) \in A \]

- Train capacity constraints:

\[ \sum_{k \in K} f_{pq}^k v^k l^k \leq L_t n_{pq}, \quad \forall p, q \in N \]

\[ t_{pq} \leq \sum_{k \in K} f_{pq}^k v^k l^k, \quad \forall p, q \in N \]

- Constants:
  - \( u_{ij} \) – Capacity of the arc \((i, j)\)
  - \( l^k \) – Length of commodity \(k\)
  - \( L_t \) – Max. allowed train length
Problem definition (cont.)

• Commodity assignment constraints:

\[ f_{pq}^k \leq s_{pq}^k + m_{pq}^k + o_{kp}, \quad \forall p, q \in N, \forall k \in K \]

\[ f_{pq}^k \leq s_{pq}^k + m_{pq}^k + d_{kp}, \quad \forall p, q \in N, \forall k \in K \]

\[ \sum_{q \in N} f_{pq}^k - \sum_{q \in N} f_{qp}^k = o_{kp} - d_{kp}, \quad \forall p \in N, \forall k \in K \]

• Constants:
  • \( o_{kp} \) – Determines if node \( p \) is the origin of commodity \( k \)
  • \( d_{kp} \) – Determines if node \( p \) is the destination of commodity \( k \)
Problem definition (cont.)

- **Node type constraints:**
  \[ r_i + s_i + m_i = 1, \quad \forall i \in N \]
  \[ \sum_{k \in K} s_i^k \leq s_i \mathcal{M}_1, \quad \forall i \in N \]
  \[ \sum_{k \in K} m_i^k \leq m_i \mathcal{M}_2, \quad \forall i \in N \]

- **Inner arc capacity constraints:**
  \[ \sum_{k \in K} v^k (s_i^k + m_i^k) = d_i, \quad \forall i \in N \]
  \[ d_i \leq s_i C_S + m_i C_M, \quad \forall i \in N \]

- **Constants:**
  - \( C_S \) – Max. shunting yard capacity
  - \( C_M \) – Max. marshaling yard capacity

- **Variables:**
  - \( r_i \) - If node \( i \) is a regular station
  - \( s_i \) - If node \( i \) is a shunting yard
  - \( m_i \) - If node \( i \) is a marshaling yard
  - \( d_i \) - The required capacity of a shunting or marshaling yard at node \( i \)
Problem definition (cont.)

- Max. number of marshaling and shunting yards:
  \[
  \sum_{i \in N} s_i \leq U_S \\
  \sum_{i \in N} m_i \leq U_M
  \]
- Constants:
  - \( U_S \) – Max. number of shunting yards
  - \( U_M \) – Max. number of marshaling yards
- Variable constraints:
  - \( t_{pq} \in \{0,1\}, \forall p, q \in N \)
  - \( n_{pq} \in \mathbb{N}, \forall p, q \in N \)
  - \( x_{ij}^{pq} \in \{0,1\}, \forall p, q \in N, \forall (i, j) \in A \)
  - \( f_{pq}^k \in \{0,1\}, \forall p, q \in N, \forall k \in K \)
  - \( s_i^k \in \{0,1\}, \forall i \in N, \forall k \in K \)
  - \( m_i^k \in \{0,1\}, \forall i \in N, \forall k \in K \)
  - \( r_i \in \{0,1\}, \forall i \in N \)
  - \( s \in \{0,1\}, \forall i \in N \)
  - \( m_i \in \{0,1\}, \forall i \in N \)
Heuristic algorithm – development details

• Developed algorithm is very flexible:
  • Easily extendable with additional neighborhood operators, i.e. network transformations
  • Easy definition of specific initial network states, e.g. all marshaling yards closed, several additional marshaling yards open, etc.