An efficient MIP model for locomotive routing and scheduling

M. Aronsson\textsuperscript{1}, P. Kreuger\textsuperscript{1} & J. Gjerdrum\textsuperscript{2}

\textsuperscript{1}Swedish Institute of Computer Science, Sweden
\textsuperscript{2}Green Cargo AB, Sweden

Abstract

This paper presents a MIP model for a locomotive routing and scheduling problem from the domain of freight railways. Innovative features of the model include the use of binary variables to separate the integer and continuous parts of the problem to maintain the flow character of the integer part of the problem. The model has been developed with, and has found practical Green Cargo, the largest rail freight operator in Sweden.

Keywords: vehicle routing and scheduling, rail traffic resource management.

1 Introduction

The increasing competition within the railway transportation sector requires effective resource utilisation methods for companies such as Green Cargo, the largest rail freight operator in Sweden.

In many countries in Europe, railroads have traditionally been state-owned organisations with diverse interests in e.g. passenger traffic, freight traffic, infrastructure and real estate investments. The Swedish state railway was deregulated in all these areas around the millennium, creating separate companies with dedicated resources. Before the deregulation, locomotives were used for passenger traffic in the daytime and freight traffic at night. Today, locomotives are dedicated to either cargo or passenger traffic, which has brought about utilisation patterns such as in figure 1.
1.1 Timetabling

Railway operators on deregulated markets have to adhere to timetables partly designed by the rail infrastructure managers. Operators apply for timed infrastructure allocation (timetable slots) based on information about traffic patterns, customer requirements, and operator resource consideration. If no slot conflicts arise, the operators normally receive their slots, but if not, they either have to accept alternative slots proposed by the infrastructure manager, or negotiate to influence an arbitration process. In this negotiation process, arguments involving customer demands and resource limitation are seriously considered by the infrastructure manager.

This paper addresses the problem of generating schedules and corresponding turnaround plans for locomotives, that have to satisfy both customer requirements and limits on operator resource utilisation costs. Resource conflicts on infrastructure resources are handled by the infrastructure manager and are out of the control of the individual traffic operator and not addressed in this paper. Similarly, vehicle maintenance requirements are handled in the fleet assignment process that uses the proposed turnaround plan as input.

1.2 Locomotive optimisation

The locomotive optimisation process determines the turnaround plan for all locomotives. In this process, a sequence of timetable slots are assigned to each vehicle such that all transports are covered by the appropriate type and number of locomotives. Transferring a locomotive from one transport to another is called a “turn”, and the set of all turns is called a turnaround plan.

Traditionally, the timetable slots for the transports are considered as given in this process. However, if the slots can be shifted in time, many turns that would otherwise be considered infeasible become possible, which can lead to a reduction in the number of locomotives required to perform the same number of transports.
2 Problem description

Minimum cost network flow models are extensively used (see e.g. [1]) to compute an optimal assignment of vehicles to scheduled transports (train movements). Transports are represented as nodes in a network. The fact that a vehicle used by one transport can also be used by another one, is represented as a (directed) arc between their corresponding nodes. Classical network flow models of this kind usually have set partitioning structure and binary flow variables so that each transport is assigned to a unique vehicle.

A straightforward generalisation of this type of flow model for cyclic schedules allows (small) integer values for the flows, and has been used for engine routing in rail transportation (see e.g. [2]). In such models, additional integer variables are associated with each node to encode how many vehicles travel with each transport. Flow is conserved on each node, giving cyclic schedules for each vehicle. Lower and upper bounds on the node variables capture the minimum and maximum number of vehicles required and usable by each transport.

Lower bounds on the node variables in the cases considered here vary from 0 on (potential) vehicle relocations to 2 for heavy freight transports. Upper bounds larger than the corresponding lower ones encode the possibility to relocate additional accompanying locomotives with a planned transport that is already served with the required number of vehicles. With a cost function penalising the total number of vehicles needed, we get a straightforward and practical model that has seen several years of practical use in e.g. the Swedish rail industry.

Normally, the network is statically generated using temporal non-overlap and distance conditions on the transports. It would be of great practical value if this kind of model could be generalised to allow for rescheduling of transports in cases where this would significantly reduce the cost of vehicle usage. However, using time windows for the departure times of the transports and an initial network with connections between any two transports that arrive and depart from the same location, breaks the locality (and hence, the network structure) of the model.

Problems of this general type are variants of the “multiple Travelling Salesman Problem” (m-TSP). The case with time windows is normally referred to as a “multiple Travelling Salesman Problem with Time Windows” m-TSPTW [1, 3–5]. This problem is normally (e.g. [6]) considered as a special case of the extensively studied class “Vehicle Routing Problems” (VRPs) [7].

The current paper presents a practical MIP model of this problem that can be used to efficiently and exactly solve practical problems up to the size of those occurring in real life transportation planning, for moderate sizes of departure time windows (< 3 hours), using a state-of-the-art commercial solver.

The model and its implementation for the solution of a large scale practical case is presented. The transports in this case form a set of train transports with a fixed schedule whose departure times are relaxed from ±15 up to ±90 minutes, and the vehicles considered are the locomotives used to pull the trains. Performance results for solving several versions of the practical problem using CPLEX 9 [8] on a PC-type workstation are also reported.
3 Model parameters

The model is parametrised by a number of constants and variables with associated bounds that will be summarised here. The constraints and objective function will be presented in section 5. Note that we have chosen to present the variable bounds in connection with their respective variables. Note also that the problem is periodic, i.e., the transport schedule is repeated after a fixed period $CT$, typically a week. The individual vehicle schedules may, on the other hand, span several such periods.

3.1 Constants

$CT$ Cycle time (period after which the transport schedule is repeated).
$t_i$ Travel time for transport $i$. We require each $t_i$ to be positive and strictly smaller than $CT$.
$p_i$ Penalty per vehicle accompanying transport $i$ above that of its vehicle requirement.
$lo_i, ld_i$ Origin and destination locations of transport $i$.
$r_{ij}$ Setup time (turn time) for the exchange of one or more vehicles between transports $i$ and $j$. We require each $r_{ij}$ to be positive and fulfil the inequality $t_i + r_{ij} < CT$.

3.2 Decision variables (discrete)

$X_{ij}$ Integer variable, determining how many vehicles are turned (transferred) from transport $i$ to transport $j$. In the practical cases considered below, the lower bound $X_{i,j}$ is normally 0, and the upper bound $\overline{X}_{i,j}$ is either 1 or 2.
$C_{ij}, C'_{ij}$ Boolean variables, used to determine if a turn from transport $i$ to transport $j$ crosses the cycle time border $CT$.
$Y_{ij}, Y'_{ij}$ Integer variables, which for any optimal solution will have the values $Y_{ij} = C_{ij}X_{ij}$ and $Y'_{ij} = C'_{ij}X_{ij}$ respectively.
$S_i$ Integer variable used to represent the number of vehicles assigned to transport $i$. A lower bound $\underline{S}_i$ on this variable encodes the minimal vehicle requirement of the transport while an upper bound $\overline{S}_i$ limits the number of vehicles usable by it.
$E_i$ Integer variable used to encode the number of vehicles accompanying a transport in addition to the number $S_i$ required by the transport itself.

3.3 Time point variables (continuous)

$d_i$ Continuous variable denoting the departure time of transport $i$. The departure time window is represented by the bounds $\underline{d}_i$ and $\overline{d}_i$ of $d_i$. 

www.witpress.com, ISSN 1743-3509 (on-line)
This formulation does not guarantee that the arrival times \( d_i + t_i \) will always be smaller than \( CT \), which influences the formulation of the constraints relating the arrival and departure events of the transports. The next section gives a case analysis of the situations that can occur, and motivates the constraint formulation given in the section following it.

4 Turning over the cycle time border

The cases are illustrated by figures where coloured bars represent the transports. The vertical extension of a coloured bar is the travel time of the transport (the interval between scheduled departure and arrival time). The surrounding transparent bar illustrates the departure time window of the transport so that the coloured bar may be placed anywhere within the transparent one.

There are four main cases for a turn from transport \( i \) to transport \( j \) to consider, each one described below.

\( A_0 \) The turn, if chosen, will never cross the cycle time border, i.e.:
\[
\overline{d_i + t_i + r_{ij}} \leq \overline{d_j}
\]

\( A_1 \) The turn, if chosen, is certain to cross the cycle time border exactly once, i.e.:
\[
(d_i + t_i + r_{ij} > \overline{d_j}) \land \left(\overline{d_i + t_i + r_{ij}} \leq \overline{d_j + CT}\right)
\]

\( A_2 \) A more rare case, which nevertheless has to be taken into account, is when the turn, if chosen, is certain to cross the cycle time border twice. Note that in this case (as well as sometimes in \( A_1 \)), two instances of the transport that crosses the border have to be considered, one leaving the period and one entering the period, i.e.:
\[
d_i + t_i + r_{ij} > \overline{d_j + CT}
\]

A turn like this is hardly ever desirable, at least not if the period time is long in comparison with the longest travel time.

A more complex case occurs when the time windows overlap so that the turn may or may not cross the cycle time border one or two times, but the exact number depends on the assignment of the departure time variables.
In the general case, it is possible to distinguish the following subcases:

**B₁** The turn may cross the cycle time border once, or not at all, i.e.:

\[
(d_i + t_i + r_{ij} \leq d_j) \land (d_i + t_i + r_{ij} > d_j) \land (d_i + t_i + r_{ij} \leq d_j + CT)
\]

**B₂** The turn may cross the cycle time border twice, but maybe only once, i.e.:

\[
(d_i + t_i + r_{ij} > d_j) \land (d_i + t_i + r_{ij} \leq d_j + CT) \land (d_i + t_i + r_{ij} > d_j + CT)
\]

**B₃** The turn may cross the cycle time border twice, once, or not at all, i.e.:

\[
(d_i + t_i + r_{ij} \leq d_j) \land (d_i + t_i + r_{ij} > d_j + CT)
\]

In the model below, we penalise the case where a turn crosses the cycle time border two times twice as hard as the case where it does so only once, which means that turns of this type are almost never found in an optimal solution.

### 5 Model constraints and objective

The cases labelled \(A₀\) through \(A₂\) above are all, if used as turns in a solution, determined to cross the cycle time limit either once, twice, or not at all. The cases labelled \(Bᵢ\), on the other hand, are indeterminate, and will be collectively encoded using the two boolean decision variables \(C_{ij}\) and \(C'_{ij}\). To be able to treat the \(A\)
and $B$ cases separately, we will define four mutually exclusive subsets of possible turns. We will need to introduce binary decision variables only for the $B_i$ cases.

Let

$A_0 = \{ (i, j) \mid 0 < i, j \leq n \land d_i + t_i + r_{ij} \leq d_j \}$

$A_1 = \{ (i, j) \mid 0 < i, j \leq n \land (d_i + t_i + r_{ij} > d_j) \land (d_i + t_i + r_{ij} \leq d_j + CT) \}$

$A_2 = \{ (i, j) \mid 0 < i, j \leq n \land (d_i + t_i + r_{ij} > d_j + CT) \}$

and

$B = \{ (i, j) \mid 0 < i, j \leq n \} \setminus (A_0 \cup A_1 \cup A_2)$

Since the main objective of the model is to minimise the number of vehicles used by a solution, and this corresponds exactly to the number of vehicles turned over the cycle time limit, the objective function will treat each of these cases (except $A_0$ which can never contribute to the cost) separately.

We also introduce a term in the cost function that penalises the use of additional vehicles for transports that do not need them. Such relocations are in most cases necessary to balance the flow of the model, but should otherwise be avoided. This penalty is weighted by the (temporal) length $t_i$ of the transport, and a factor $p_i$ specific to each transport.

Minimise

$$\sum_{(i, j) \in A_1} X_{ij} + \sum_{(i, j) \in A_2} 2X_{ij} + \sum_{(i, j) \in B} (Y_{ij} + Y'_{ij}) + \sum_{0 < i \leq n} E_i p_i t_i$$

subject to

1. The number of vehicles turned from transport $i$ is equal to the number used by it

$$\forall i ((\sum_{j \in \{ j \mid ld_i = lo_j \}} X_{ij}) - S_i = 0)$$

and the number of vehicles turned to transport $j$ is equal to the number used by it

$$\forall j ((\sum_{i \in \{ i \mid ld_i = lo_j \}} X_{ij}) - S_j = 0)$$

The candidate turns are chosen so that the destination $ld_i$ of the source transport $i$ and the origin $lo_j$ of the sink $j$ are identical.

2. Turn time constraints

$$\begin{cases}
  d_j - d_i + CT C_{ij} + CT C'_{ij} > t_i + r_{ij} \\
  X_{ij} - Y_{ij} + M C_{ij} \leq M \\
  X_{ij} - Y'_{ij} + M C'_{ij} \leq M
\end{cases} \quad \forall i, j ((i, j) \in B)$$

and

$$S_i - E_i = S_i \quad \forall i$$

(a) $C_{ij}, C'_{ij}$ boolean, $C'_{ij} \leq C_{ij}$

(b) $S_i, X_{ij}$ (implicitly) integer

(c) Variable bounds $d_i \leq d_i \leq d_i, S_i \leq S_i, X_{ij} \leq X_{ij} \leq X_{ij}$ for $\forall i, j$
5.1 Constraint notes

The flow (conservation) constraints (1) ensure that each transport is supplied with as many vehicles as it needs and that the flow is balanced. To ensure that this is always possible, we need to introduce vehicle relocations. This is done by adding “passive” transports with a vehicle requirement of zero, that need not be performed unless they are assigned at least one vehicle. These are penalised more severely than additional locomotives assigned to “real” transports.

The turn time constraints (2) and their use of the boolean variables (2a) are the core of the model. Note that, for any optimal solution, \( C_{ij} = C'_{ij} = 0 \) if and only if \( d_i + t_i + r_{ij} \leq d_j \), that \( C_{ij} = 1 > C'_{ij} \) if and only if \( d_j < d_i + t_i + r_{ij} \leq d_j + CT \), and finally that \( C_{ij} = C'_{ij} = 1 \) if and only if \( d_i + t_i + r_{ij} > d_j + CT \), corresponding exactly to the three A-cases above. This follows from the fact that unnecessarily assigning 1 to \( C_{ij} \) while \( X_{ij} > 0 \), will be penalised by forcing \( Y_{ij} \) to become equal to \( X_{ij} \), and similarly for \( C'_{ij} \) and \( Y'_{ij} \).

A key feature of the model, and the main reason that it scales relatively well in practise, is that the integrality constraints on \( S_i \), and \( X_{ij} \) (2b) need not be enforced by the solver. In each leaf in the search tree branching on the boolean variables \( C_{ij} \) and \( C'_{ij} \), the part of the coefficient matrix involving these variables will be a pure minimal cost flow. The same obviously does not apply to the part involving the departure time variables \( d_i \), but since these variables are related to the decision variables \( S_i \) and \( X_{ij} \) only through the booleans \((C_{ij}, C'_{ij})\), each assignment of the \( d_i \) variables that is consistent with a complete (integral) assignment of the booleans, will also be consistent with the optimal assignment of the decision variables \( S_i \) and \( X_{ij} \). This means that the optimal solution to the problem obtained by relaxing the integrality constraints on \( S_i \) and \( X_{ij} \) (but not on \( C_{ij} \) and \( C'_{ij} \)) will also be an optimal solution to the original problem.

6 Empirical results

The performance results have all been produced using data from the largest Swedish rail freight company Green Cargo. The case consists of 1304 transports and contains almost all transports handled by their most common vehicle, the electrical RC locomotive, for one full week. The problems solved below were generated by introducing a fixed amount of slack for each departure time in the production plan.

In the solutions reported below, accompanying locomotives have been freely introduced and moved around between transports that allow them. Passive transports, on the other hand, are eliminated wherever this leads to an improved objective.

Note that introducing slack uniformly is not completely realistic. In reality, customer requirements or limits on infrastructure capacity may not allow free rescheduling of transports within their time windows. To some extent, this can be improved by introducing individual slack for each transport, and weighted binary relations between arrival and departure events that encode e.g. transfers of cars.
Table 1: Without additional passive transports.

<table>
<thead>
<tr>
<th>Slack minutes</th>
<th>Booleans</th>
<th>Vehicles</th>
<th>Accompanying minutes</th>
<th>Deviation minutes</th>
<th>Run time h:mm:ss</th>
</tr>
</thead>
<tbody>
<tr>
<td>±0</td>
<td>-</td>
<td>117</td>
<td>50835</td>
<td>-</td>
<td>0:00:05</td>
</tr>
<tr>
<td>±15</td>
<td>1027</td>
<td>116</td>
<td>50206</td>
<td>5107</td>
<td>0:00:07</td>
</tr>
<tr>
<td>±30</td>
<td>1995</td>
<td>112</td>
<td>51107</td>
<td>13763</td>
<td>0:00:54</td>
</tr>
<tr>
<td>±45</td>
<td>2836</td>
<td>105</td>
<td>51177</td>
<td>20841</td>
<td>0:01:19</td>
</tr>
<tr>
<td>±60</td>
<td>3913</td>
<td>99</td>
<td>49402</td>
<td>35651</td>
<td>0:08:22</td>
</tr>
<tr>
<td>±75</td>
<td>4930</td>
<td>97</td>
<td>49411</td>
<td>48486</td>
<td>1:53:40</td>
</tr>
<tr>
<td>±90</td>
<td>5876</td>
<td>90</td>
<td>50385</td>
<td>69067</td>
<td>23:22:43</td>
</tr>
</tbody>
</table>

and cargo. In the performance results reported here, no such additional constraints were used. Nevertheless, a production version of the software used to generate these problems is currently in use at Green Cargo in their planning of locomotives.

Table 1 reports, for each slack size (in minutes), the number of booleans needed to encode the turn time constraints, which should give a rough indication of the MIP size. It also reports properties of the optimal solution found in terms of the number of vehicles, the total amount of accompanying locomotives, and (additional) “passive” time in minutes. Performance results in terms of run time in seconds for each slack size are also included. More specifically, the run times are those reported by CPLEX 9 on an 2.4 GHz Pentium 4 processor using about 2 GB of main memory. For the larger cases, caching the node tree to disk was done whenever it became larger than the main memory. The strategy used was the default heuristic of CPLEX 9 [8].

Once the optimal solution for the locomotive turns has been found, a new timetable is generated minimising the sum of deviations from the original timetable. This problem is linear and no performance results of these runs are given. The resulting deviation (in minutes) is given in the table to give an indication of how much the original timetable had to be changed to achieve the corresponding improvement of the main objective.

As can be seen from the tables, the number of booleans increase more or less linearly with increased time window size, which is reasonable since the booleans correspond to temporal overlaps between transports potentially served by the same vehicle. The number of vehicles used by the optimal solutions to the relaxed timetable is also substantially reduced for increased time window sizes. Going e.g. from 117 locomotives to 90 represents a reduction of vehicle usage by 23%, which would be sensational were it not for the fact that the current model does not take track slot availability into account. Still, these figures do show the potential of taking locomotive fleet costs into account when planning the timetable.
Run times increase fairly rapidly with time window size, but none of the cases above are completely unrealistic for a practical workflow. The model is not particularly sensitive to different timetables, but the potential gain obviously varies.

7 Conclusions

We show how rescheduling transports can reduce locomotive usage in a train transportation problem. An optimisation model for transport departure time windows varying around an initial target time is formulated and it is shown that substantial reduction of locomotive usage (up to 23%) can be achieved using a standard commercial IP-solver.

Innovative features of the model include the use of boolean variables to separate the integer and continuous parts of the problem to maintain the flow character of the integer part of the problem for each complete assignment of the booleans.

Application of the model produces a modified train schedule that accommodates the requirements for an efficient locomotive turnaround plan. The practical usefulness of the model and its scalability is demonstrated on a set of problems derived from a real case in the Swedish rail freight industry.

Significant savings can be realised for a uniform fleet of locomotives, in terms of locomotives planned, by utilising the presented method.

References