MINOA Challenge - 2021

Georgina McCosh, Ziyad Benomar

March 15, 2021

Introduction

Project Aim

To minimise cost in an Integrated Timetabling (TT) and Vehicle Scheduling (VS) Problem

Motivation

- Integrated TT-VS problem has better potential to minimise costs over traditional methods
- Deployment of electric vehicles in public transport systems is increasing globally
- The health crisis has highlighted the importance of being able to quickly adapt PTN planning

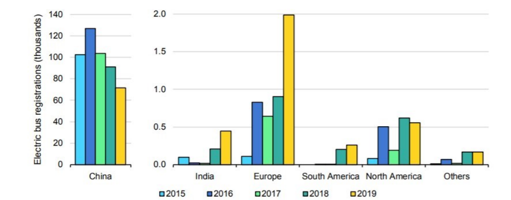

Figure: Electric bus registrations. Source: IEA 2020

Problem Description

- Nodes are connected by lines
- Each direction has a main stop
- Each main stop has a maximum headway
- Potential trips are given in the data
- Deadhead arcs are trips which serve no passengers
- The data was rewritten from a json file supplied into the format we needed

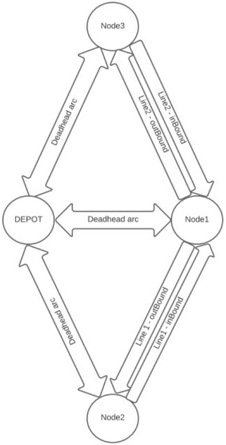

Figure: Diagram of lines and nodes. Source: MINOA Challenge Description

Model

Model : VS Constraints

- Electric/ICE vehicles ?
- Number of ICE vehicles ?
- Decision variables ?
- Constraints on "consecutive" trips ?
- ...

Assumptions

Only two types of vehicles : $v_b \in \{$"ICE", "electric"$\}$
finite number of ICE vehicles
default : 2*(number of electric vehicles)

- Input : (Num of ICE vehicles) + (Num of electric vehicles)
- Sets of vehicles

$$BE := \{1..(\text{Num\_ele})\}$$
$$BI := \{(\text{Num\_ele}) + 1..(\text{Num\_ice})\}$$
$$B := BE \cup BI = \{1..(\text{Num\_ice})\}$$

Model : VS Constraints

Consecutive trips

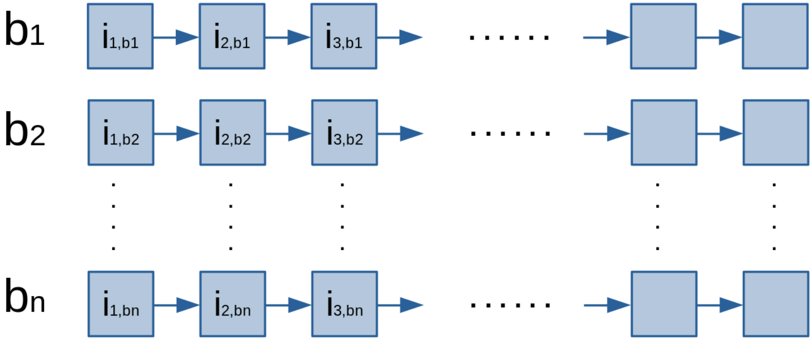

Figure: Vehicule scheduling

- Elementary decision variables : for $b \in B, i \in T, j \in$ **?**

  $x[b,j,i] = 1 \iff i$ is the trip number j in the block b

- Maximum number of trips in each block ? $|T|$!
- For $b \in B, j \in \{1..|T|\}$, let

$$y[b,j] := \sum_{i \in T} x[b,j,i]$$

- $y[b,j]$ must be a binary variable

$$\begin{aligned} y[b,j] = 1 &\iff \text{There is a trip number } j \text{ in the block } b \\ &\iff \text{The block } b \text{ contains at least } j \text{ trips} \end{aligned}$$

$$(y[b,j])_{j\in\{1..|T|\}} = (1,1,0,1,0,0,1,0,1,...) \text{ NO !}$$

$$(y[b,j])_{j\in\{1..|T|\}} = (1,1,...,1,1,0,0,,...,0) \text{ YES !}$$

Constraints defining $x[b,j,i]$ and $y[b,j]$

- $x[b,j,i], y[b,j] \in \{0,1\} \qquad \forall b \in B, j \in \{1..|T|\}, i \in T,$
- $y[b,j] = \sum_{i\in T} x[b,j,i] \qquad \forall b \in B, j \in \{1..|T|\},$
- $y[b,j+1] \leq y[b,j] \qquad \forall b \in B, j \in \{1..|T|-1\}.$

- When is a trip i scheduled in a block b ?

$$\textsc{Sch}[b,i] := -2 + \sum_{j \in \{1..|T|\}} (j+2)x[b,j,i]$$

- Are the trips i_1, i_2 consecutive in the block b ?

$$\textsc{Consec}[b,i_1,i_2] = 1 \iff \textsc{Sch}[b,i_2] - \textsc{Sch}[b,i_1] = 1.$$

How can we formally define $\textsc{Consec}[b,i_1,i_2]$?

Absolute value of a variable

Let ζ be some variable. We can define $|\zeta|$ by introducing two additional variables ζ^+, ζ^- and imposing the following constraints

- $\zeta^+, \zeta^- \geq 0,$
- $\zeta^+ \times \zeta^- = 0$
- $\zeta = \zeta^+ - \zeta^-,$
- $|\zeta| = \zeta^+ + \zeta^-,$

$$\textsc{Consec}[b, i_1, i_2] = 1 \iff (\textsc{Sch}[b, i_2] - \textsc{Sch}[b, i_1] - 1) = 0.$$

Constraints defining Consec

Let M be a sufficiently large constant ($M = |T|$ for exple)

- $1 - \textsc{Consec}[b, i_1, i_2] \leq |\textsc{Sch}[b, i_2] - \textsc{Sch}[b, i_1] - 1|$
- $(1 - \textsc{Consec}[b, i_1, i_2]) * M \geq |\textsc{Sch}[b, i_2] - \textsc{Sch}[b, i_1] - 1|$

Summary

- Is the trip i the j-th trip in the block j ? : $x[b, j, i]$
- Does the block b contain more than j trips ? : $y[b, j]$
- When is the trip i scheduled in the block b ? : $\textsc{Sch}[b, i]$
- Are the trips i_1, i_2 consecutive in the block b ? : $\textsc{Consec}[i_1, i_2, b]$
- We can write the In-line/Out-line compatibility constraints !

Note : It is possible not to define the variables $y[b, j]$ and $\textsc{Sch}[b, i]$ because these are linear combinaisons of the variables $x[b, j, i]$

Model : VS Constraints

Writing the VS constraints

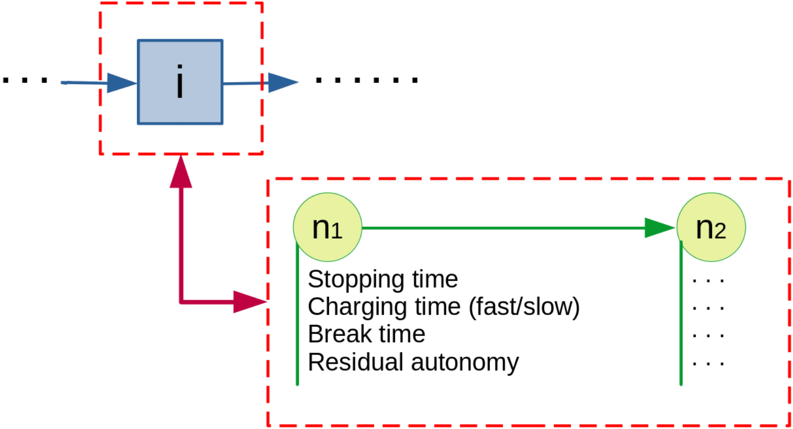

Figure: start node and end node of a trip i, and the associated variables

Model : VS Constraints

Writing the VS constraints

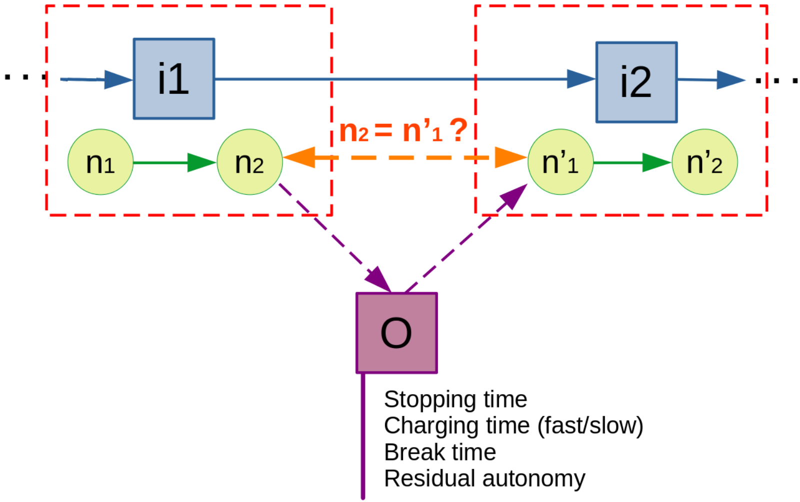

Figure: Eventual back to depot between two trips

Model : VS Constraints

Writing the VS constraints

```
# Variables for the first node of each trip in each block
var t_stop1{B,indTS} integer, >=0, default 0; # stop time at the first node of the trip
var t_break1{B,indTS} integer, >=0, default 0;
var t_fast1{BE,indTS} integer, >=0, default 0; # fast charging time at the first node of the trip
var t_slow1{BE,indTS} integer, >=0, default 0;
var recharge1{BE,indTS} binary, default 0; # deciding to recharge or not
var a_res1{BE,indTS} >=0; # residual autonomy when arriving to the node
# Variables for the second node of each trip in each block
var t_stop2{B,indTS} integer, >=0, default 0;
var t_break2{B,indTS} integer, >=0, default 0;
var t_fast2{BE,indTS} integer, >=0, default 0;
var t_slow2{BE,indTS} integer, >=0, default 0;
var recharge2{BE,indTS} binary, default 0;
var a_res2{BE,indTS} >=0; # residual autonomy when arriving to the node
# Variables for an eventual back to the depot before a trip
var t_stop0{B,indTS} integer, >=0, default 0;
var t_break0{B,indTS} integer, >=0, default 0;
var t_fast0{BE,indTS} integer, >=0, default 0;
var t_slow0{BE,indTS} integer, >=0, default 0;
var recharge0{BE,indTS} binary, default 0;
var a_res0{BE,indTS} >=0, default 0;
# Variables to decide if we go the depot before a trip or not
var back_depot{B,indTS} binary; # decides if the bus goes to the depot before its j-th trip
```

Figure: Defining the variables in the mod file

We write the relations between these variables, and we multiply them by back_depot[b,j] or (1-back_depot[b,j]) to activate/deactivate them.

Model : TT Constraints

The variables

- Is the trip i the j-th trip in its direction ? : $x_{TT}[j,i]$
- Are more than j trips planned in the direction (n_1, n_2) ? : $y_{TT}[n_1, n_2, j]$
- When is the trip i scheduled in its direction ? : $\pi[i]$
- Do trips i_1, i_2 have the same direction ? and are they consecutive in this direction ? : $\textsc{Consec}_{TT}[i_1, i_2]$

Remarks

- The direction of a trip i is given by : (sn[i],en[i])
- $x_{TT}[j,i]$ is defined for $i \in T, j \in \{1..|T_{(sn[i],en[i])}|\}$

The first trip in each direction must be in the set T_{ini}

$$\forall n_1, n_2 \in V, \quad \sum_{i \in T_{ini}(n_1,n_2)} x_{TT}[1,i] = 1$$

The last trip in each direction must be in the set T_{fin}

- count performed trips in each direction :
  $\text{count\_trips}[n_1, n_2] = \sum_j y[n_1, n_2, j]$
- determine the last trip in each direction :
  $\text{is\_last}[n_1, n_2, i] = 1 \iff \text{count\_trips}[n_1, n_2] = \pi[i]$
- $\forall n_1, n_2 \in V, \quad \sum_{i \in T_{fin}(n_1,n_2)} \text{is\_last}[n_1, n_2, i] = 1$

Model : Linking Constraints

$$\textsc{Performed}[i] = \sum_{b \in B} \sum_{j \in \{1..|T|\}} x[b,j,i],$$

$$\textsc{Performed}[i] = \sum_{j \in \{1..|T|\}} x_{TT}[j,i];$$

Solution Approach

Solution Approach

Methods in the literature

- Schmid and Ehmke [2015] decompose the problem into TT and VS stages and use a large neighbourhood search
  - LNS swaps consecutive trips between vehicles in the VS stage
- Ropke and Pisinger [2006] use iterations of destroy and repair operators to solve a similar problem
  - Schmid and Ehmke [2015] propose 3 types of destroy/repair operators: random, greedy and biased

Solution Approach

Our Method

1. Decompose the problem into TT and VS stages after an initial time-limited solution was found
2. Use a random destroy/repair operator to add and remove trips from the selected subset
    - Ignoring initial and final trips
3. Solve with fixed trip subset
4. Repeat for N iterations

This is not as sophisticated a method as proposed by Schmid and Ehmke [2015] because the local search does not have a criterion to test for improvement. Also we could use a different criteria for the destroy/repair operator.

Conclusion

Conclusion

- Data needed to be converted to the correct format
- The model : "mathematically" correct, but not efficient
- Deep study of the literature
- Difficulties to use relaxation methods or greedy algorithms
- A heuristic algorithm inspired from the course
- model with only compatibility constraints ?