An optimisation framework for determination of capacity in railway networks

Lars Wittrup Jensen

CASPT 2015 Rotterdam

RobustRailS

DTU Transport
Department of Transport

Agenda

1) Introduction

2) Optimisation framework for determining railway capacity

3) Capacity consumption in railway networks

4) Optimisation framework continued

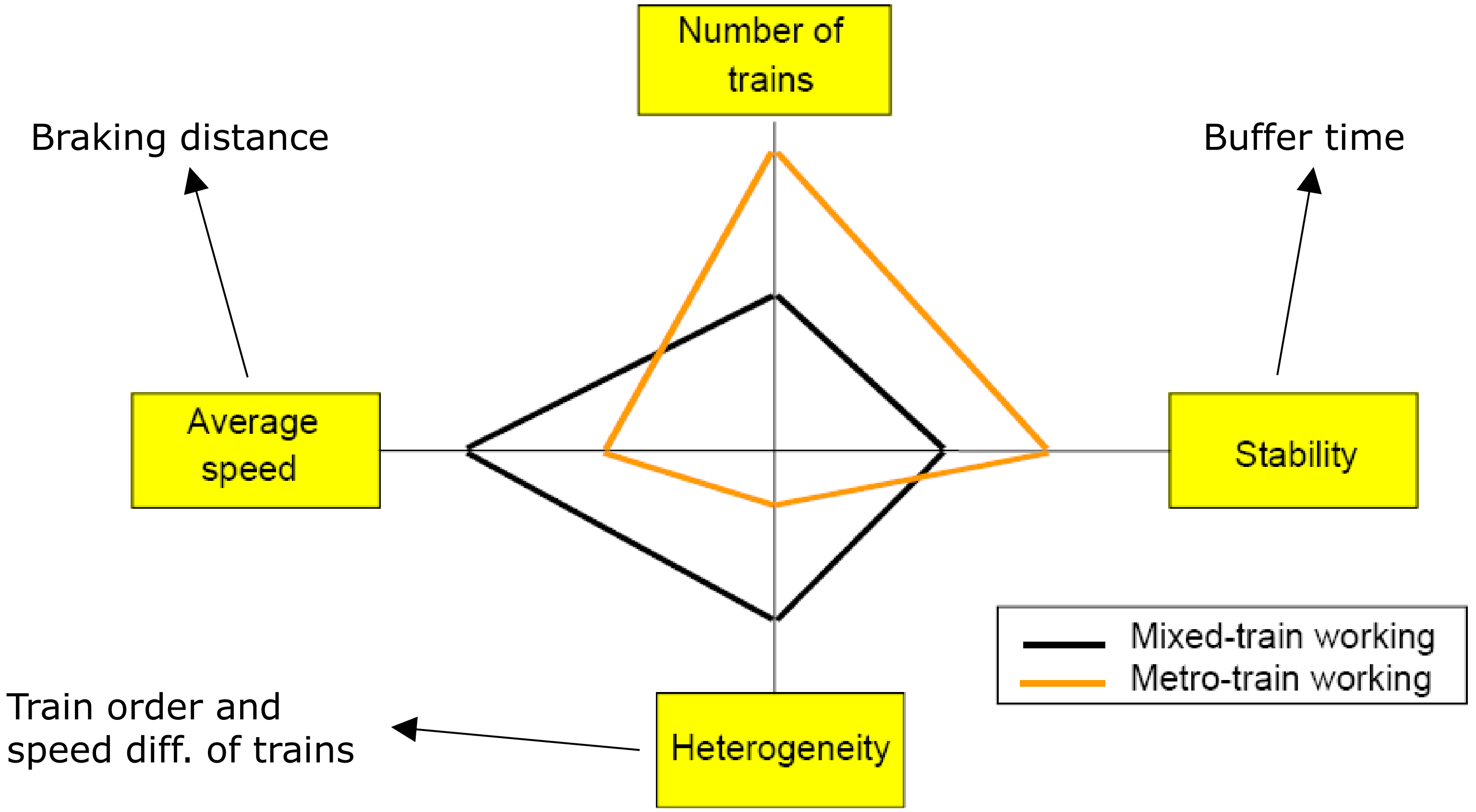
5) Case
    1) Computational results
    2) Capacity results

6) Solution space

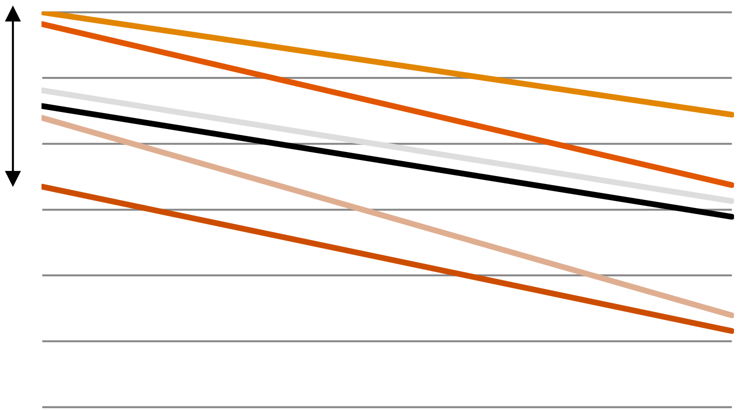
7) Conclusions and further work

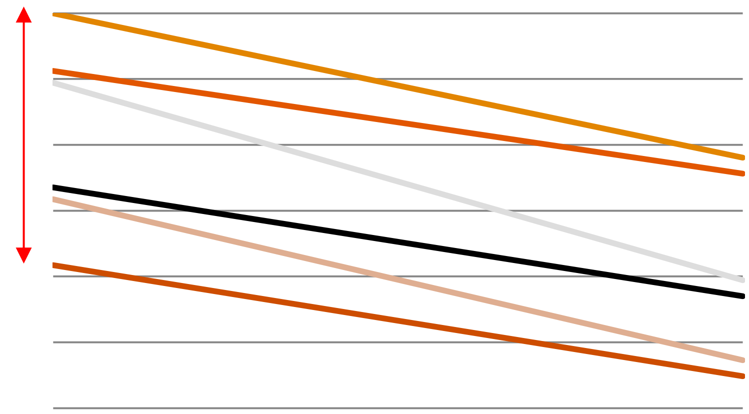
What is railway capacity?

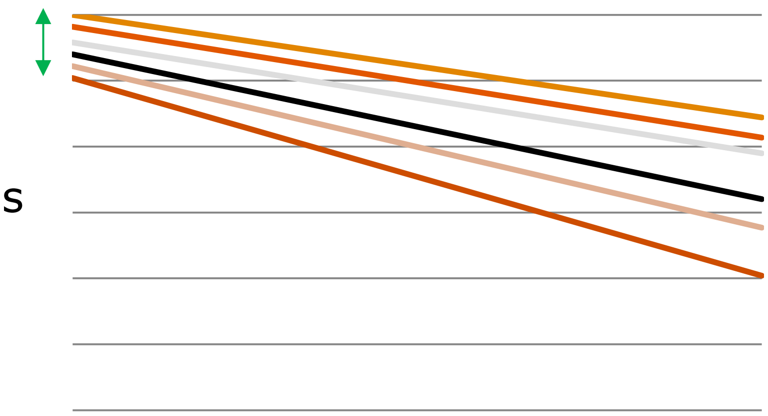
"Capacity as such does not exist. Railway infrastructure capacity depends on the way it is utilised." (UIC, 2004)



How to mix trains - Heterogeneity







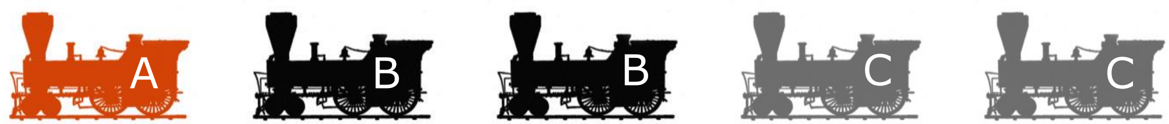
- n! for acyclic timetables
- (n-1)! permutations for cyclic timetables
- If more than 1 copy of a train type (k):
  - $n!/(k_1!k_2!\ldots k_m!)$
    (Multinomial coefficient)

Objective

- *What is the number of trains that a railway network can handle under a given robustness threshold?*
- Strategic planning: no train sequence/timetable is given beforehand
- As railway capacity depends on the sequence of trains,
  a span of number of trains must be found rather than a single value

Approach for determining railway capacity

- A train mix is given, e.g.
  - 20% of type A, 40% of type B and 40% of type C

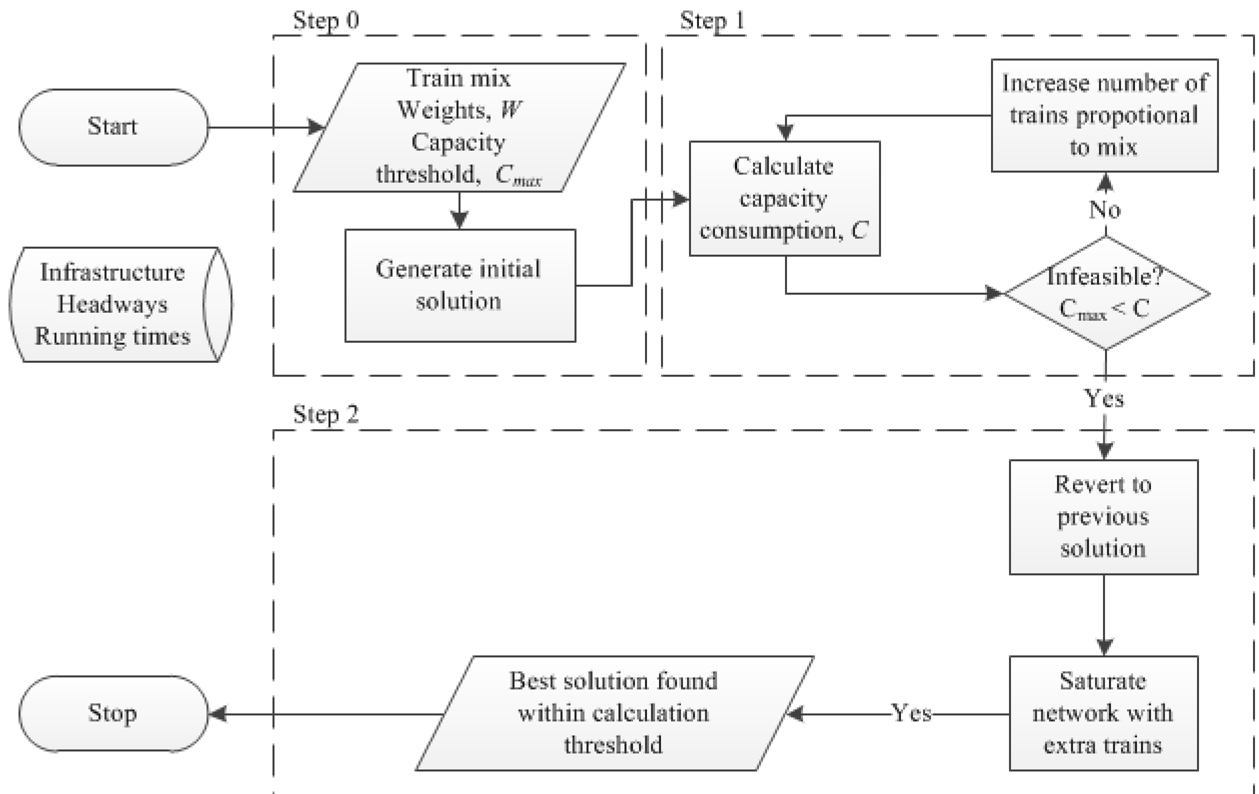


- The output is trains that can pass the network
  - According to mix, e.g.

  - Extra trains that saturate the network according to weights, e.g.

Framework

Optimisation framework:



Measuring capacity consumption

- The model/method used for capacity consumption measurement must fulfil the following requirements:
  - If an extra train is added to a network:

    The capacity consumed will never decrease compared to the previous solution (without the extra train)
  - Likewise, if a train is removed the capacity consumed will never increase

- If the span of capacity must be obtained, the model has to account for this as well (heterogeneity)

Our capacity consumption model

Input

- The network
  - Infrastructure
  - Headways
- Train type runs
  - Route
  - Running time
  - Number of each type

Calculate

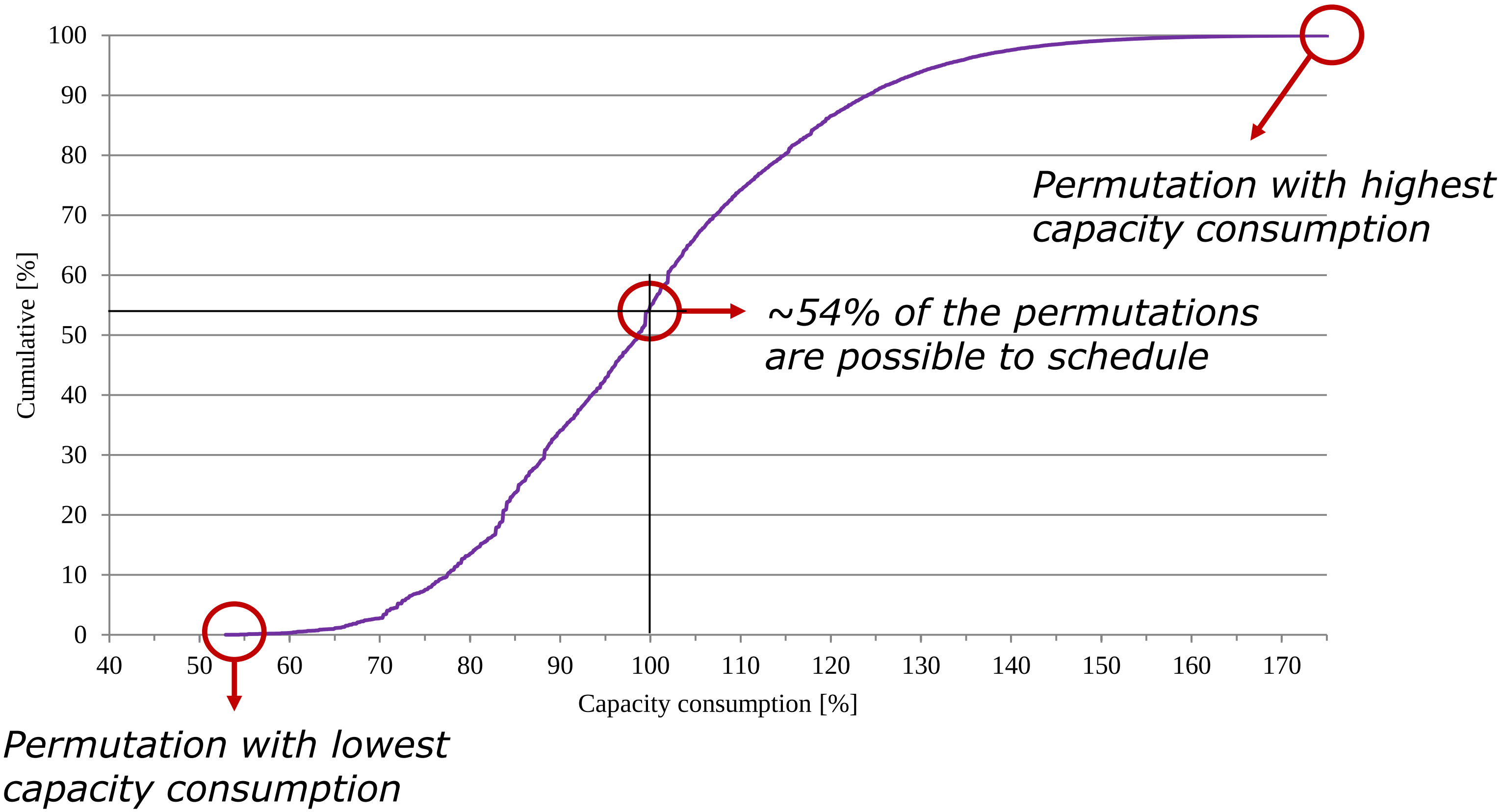
Output

- (Distribution of) capacity consumption

Calculate capacity consumption for each (random) permutation

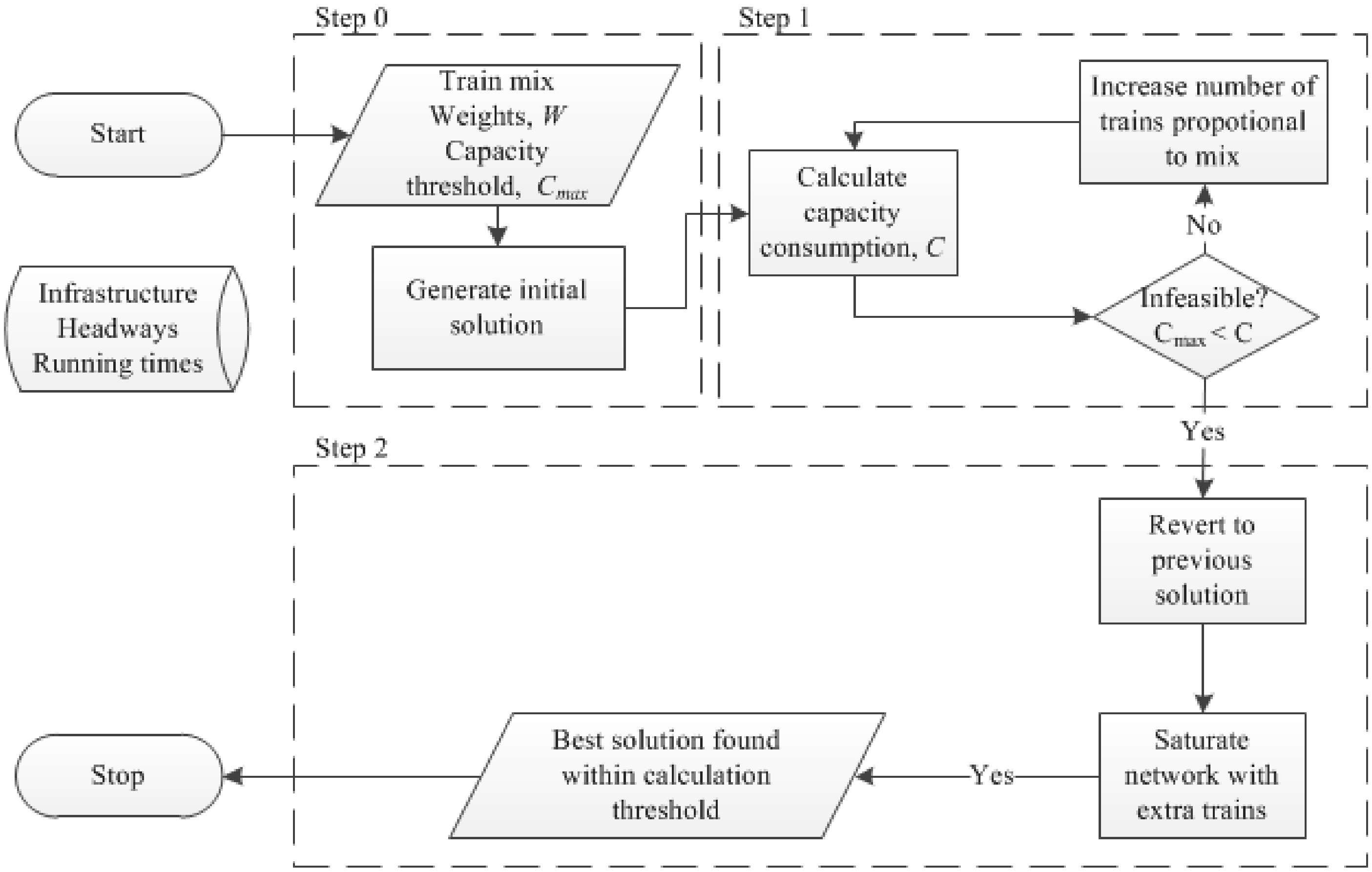
Deterministic capacity consumption

- Output
  - Cumulative distribution of capacity consumption



Framework

Optimisation framework:



Obtaining capacity span

- For the capacity consumption calculation, *C*, a percentile, *p*, is required

- Can be done for different values of *p* to obtain a span of trains:
  - 0: maximum number of trains (can only be fitted in perfect order)
  - 50: median number of trains (only for 50% of the sequences)
  - 100: minimum number of trains (can always be fitted)

Mathematical model for step 2

- Step 2: We want to add as many extra trains as possible

$$\text{maximize} \sum_{i=1}^{n} (w_i \cdot x_i)$$

s.t.

$$x_i \geq 0$$

$$C(X+Y,p) \leq C_{max}$$

Number of trains of type *i*

Weight for train type *i*

Capacity consumption of solution must not exceed threshold

$X=(x_1 \ldots x_i \ldots x_n)$

$Y=(y_1 \ldots y_i \ldots y_n)$, trains added in step 1

Percentile, *p*

p = 0: only the "best" sequence

p = 100: all sequences fit

Determining absolute railway capacity

- Simple problem, except:
  - *C(X+Y,p),* the capacity consumption, is expensive to evaluate

- Well-known within simulation-based optimisation
  - Simulation can be approximated using e.g. Response Surface Model based on regression or neural networks
  - A good approximation will provide quality solutions fast

- However, in simulation-based optimisation simulation is used for the objective function not the constraints

Solution method

- Model solved using a greedy heuristic
- A modified binary search is used to find the maximum number of trains to add of a type (*dSearch* in the code below)

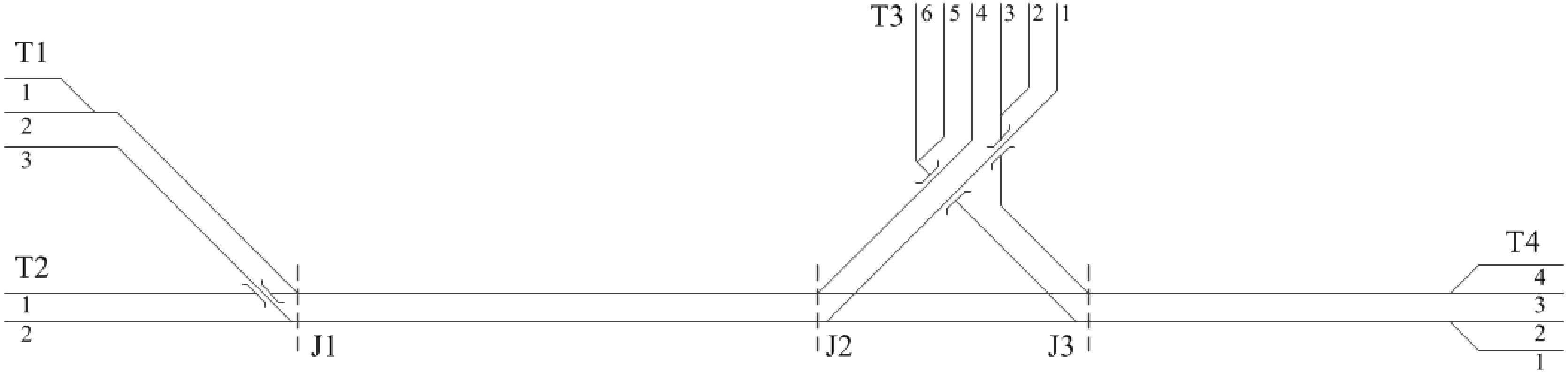
Algorithm 1: Pseudocode for greedy heuristic.

Data: Weights, $w_i \in X$
Result: $X^* = (x_1, ..., x_i, ..., x_n)$ a solution that maximize $\sum_{i=1}^{n}(w_i \cdot x_i)$
$x_i^* = 0 \quad \forall i \in \{1, ..n\}$
for $i \in X$ *in descending order of* w_i **do**
  $x_i = 0 \quad \forall i \in \{1, ..n\}$ `// Reset solution`
  $x_i = dSearch(i)$ `// Find maximum amount of trains to add`
  for $j \in X$ *in descending order by* w **do**
    if $j \neq i$ **then**
      $x_j = dSearch(j)$
    end
  end
  $X^* = max(obj\_val(X), obj\_val(X^*))$ `// Set best solution`
end
return X^*

Case

- Mesoscopic network (feasibility by microscopic input data)

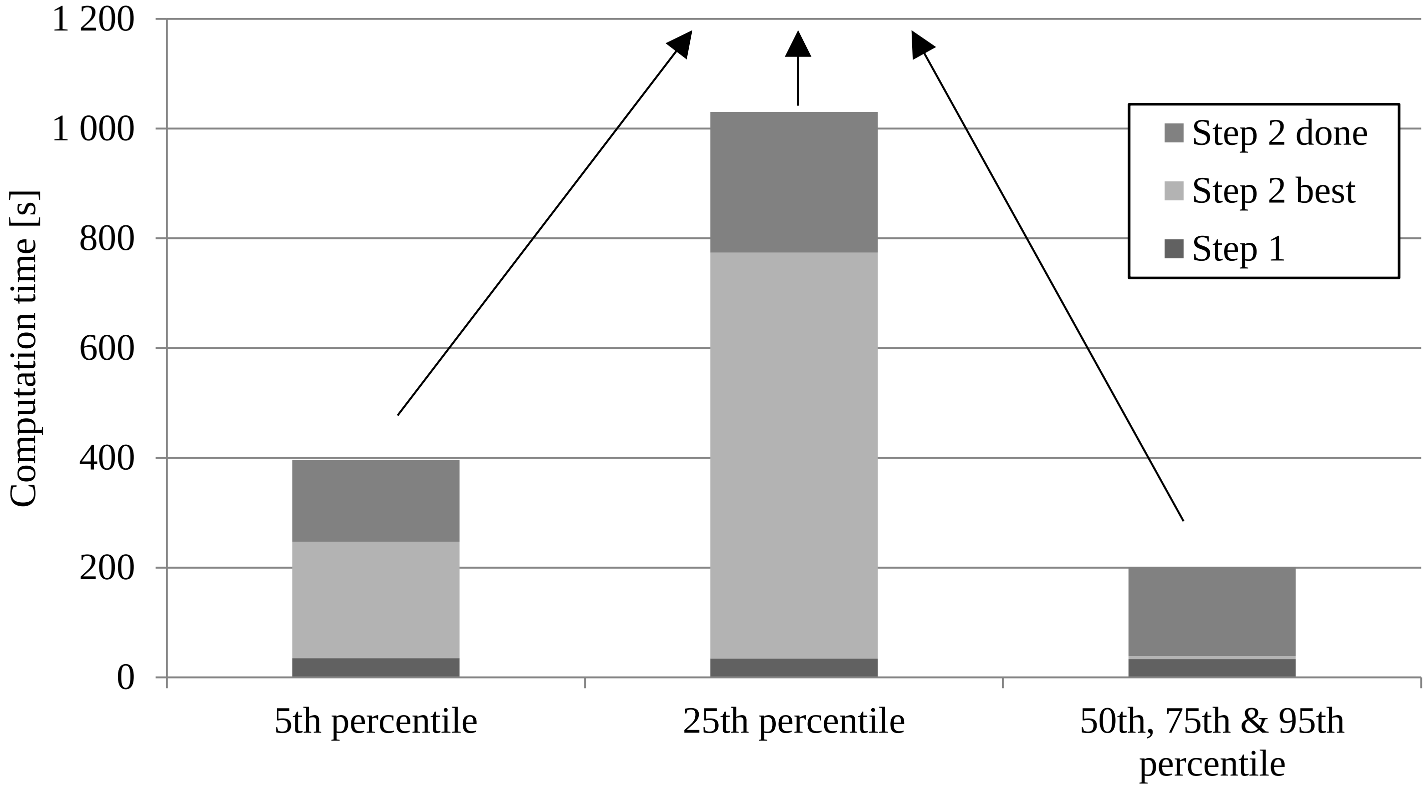
Train type	Share (step 1)	Weight (step 2)
ICE (T3↔T4)	12.5%	3
IC (T3↔T4)	12.5%	2
IC (T1↔T4)	12.5%	2
RE (T1↔T3)	12.5%	1
RE (T3↔T4)	25%	1
Freight (T2↔T4)	25%	4



Results

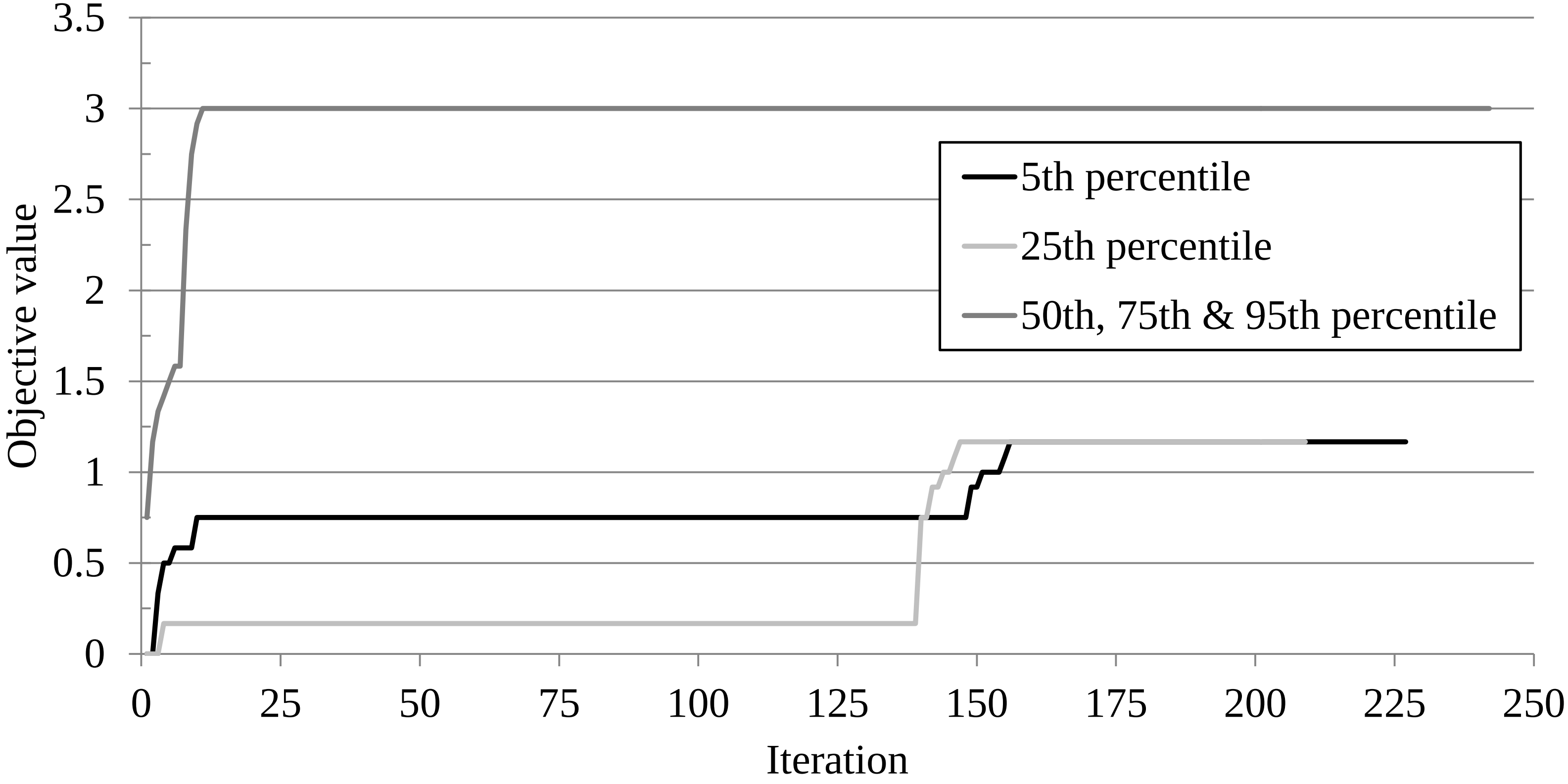
- Computational results for step 1 and 2:
  - Win 7 desktop w/ Intel Core i7-2600 (3.4 GHz) CPU and 8 GB RAM

Increased running time of *C(X+Y,p)* due to heterogeneity



Results

- Computational results for step 2



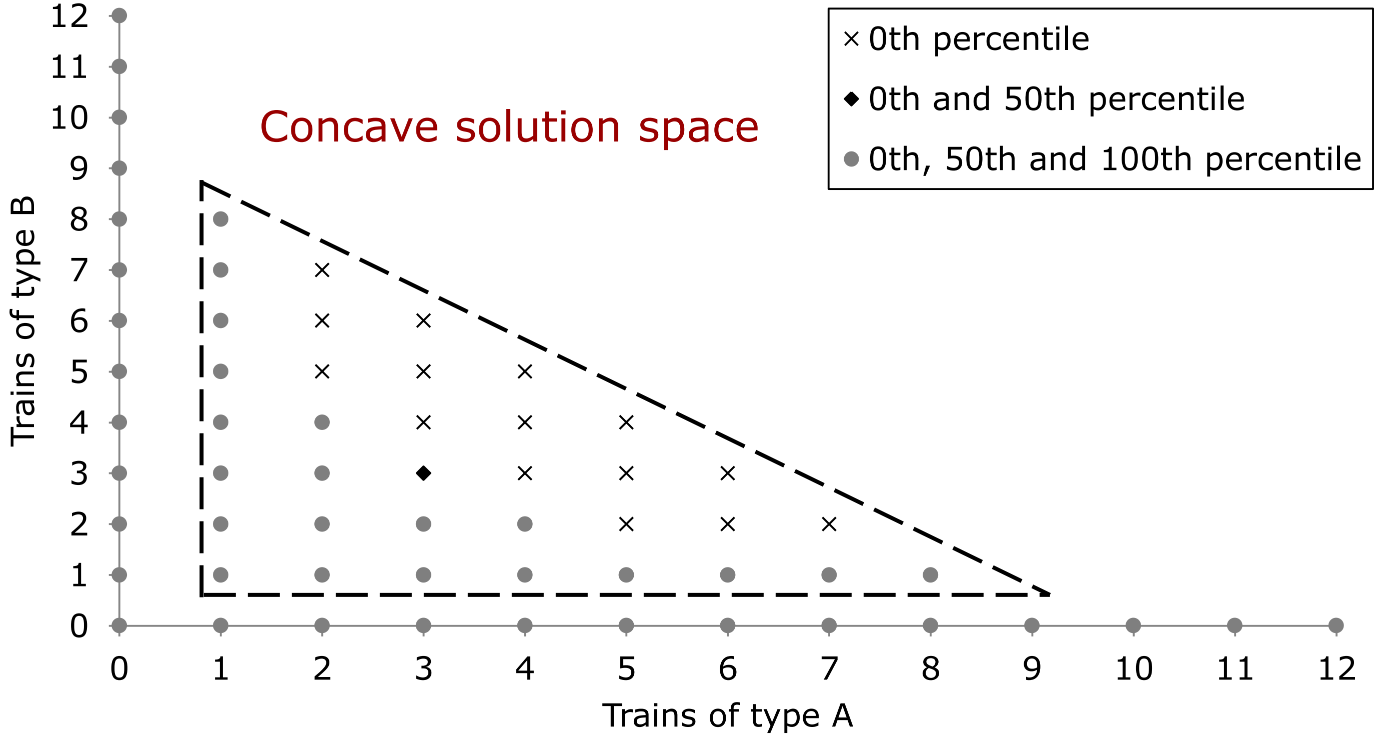
Results

- Results for case

C_{max}	75%	90%	90%	90%	100%	100%	100%
Percentiles	5-95th	50-95th	25th	5th	75-95th	50th	5-25th
Mix scheduled			•	•		•	•
Train type	Number of trains (extra trains added in step 2)						
1	0	0	1	2 (1)	0	1	2 (1)
2	0	0	1	1	0	1	1
3	0	0	1	1	0	1	1
4	0	0	1	1	0	1	1
5	0	0	1	1	0	1	1
6	0	0	1	1	0	1	1
7	0	0	1	1	0	1	1
8	0	0	1	1	0	1	1
9	0	0	16 (14)	2	0	18 (16)	2
10	0	0	2	2	0	2	2
11	16 (16)	19 (19)	2	9 (7)	22 (22)	2	11 (9)
12	18 (18)	21 (21)	2	5 (3)	24 (24)	2	7 (5)
Mix - objective	0 - 5.2	0 - 6.2	1 - 0.5	1 - 1.7	0 - 7.1	1 - 0.6	1 - 2.3

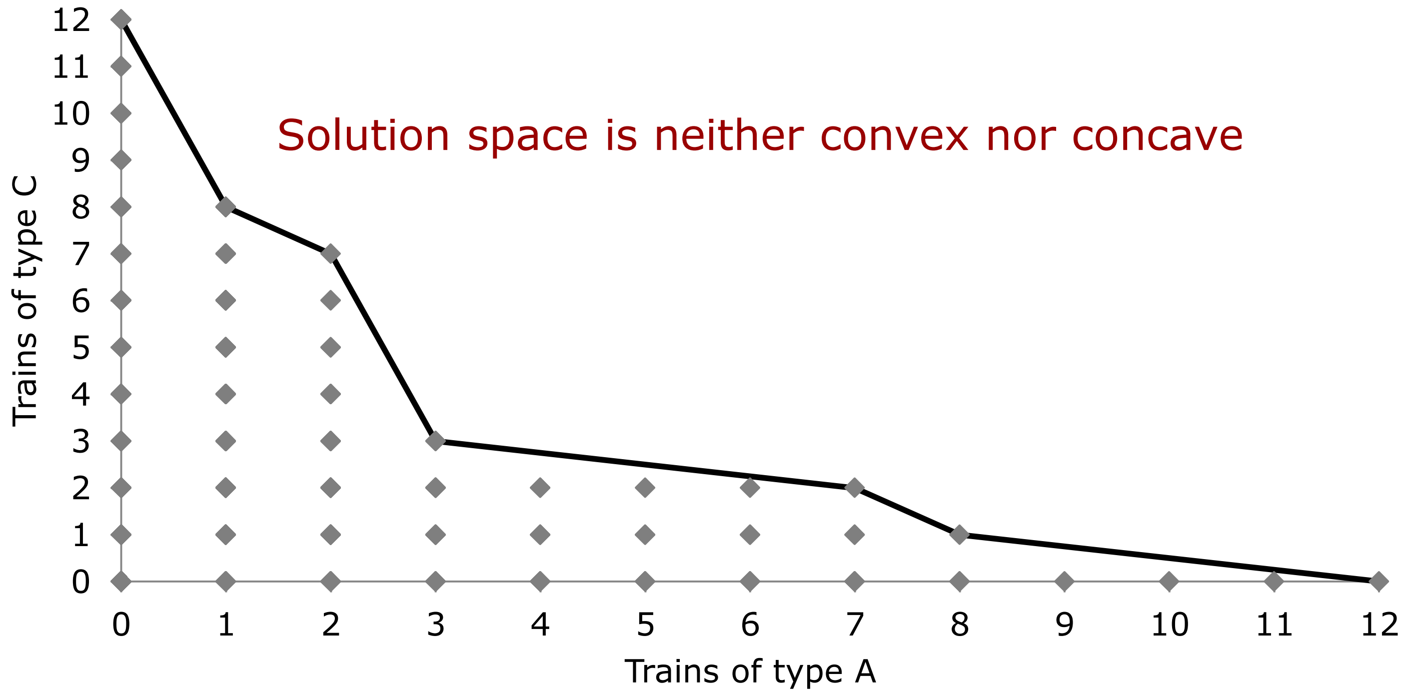
Solution space

- Does the greedy heuristic finds the optimal solution?
  - In some cases yes:



Solution space

- Does the greedy heuristic finds the optimal solution?
  - In some cases yes:
  - And in some cases no:



Conclusions

- Proposed framework determines the **number of times a certain mix of train can be scheduled plus additional trains** that saturate the mix solution using predefined train type weights

- The framework is able to give a span of capacity in networks with heterogeneous operation

- For the case considered
  - Capacity estimated in ~3 minutes (in most test instances), but in some instances up to ~20 minutes

- In step 2, as many trains as possible is added of one single type
  - Not desirable as solutions might contain a very high proportion of one train type

- Greedy heuristic does not guarantee optimal solutions

Further work

- Other solutions methods for the mathematical model (step 2)
  - Use metaheuristics to obtain better solutions

- Alternative/extended formulation of the mathematical model
  - To ensure more diversity in train types added in step 2 (not only one type)
  - To ensure directional symmetry

- Use stochastic capacity consumption model instead of deterministic
  - Will capture the aspect of robustness better and thus provide better capacity results
  - Requires some improvements in calculation time

Thank you for your attention!

Lars Wittrup Jensen
Department of Transport
Technical University of Denmark
lawj@transport.dtu.dk