DEALING WITH CONFLICTING TRAINS

Effectively avoiding and resolving conflicts while shunting trains overnight

AUTHORS

Mees Gribnau

m.gribnau@student.tudelft.nl

AFFILIATIONS

EEMCS, Delft University of Technology, The Netherlands

01 BACKGROUND

- Excess trains at a train station have to be stored in a shunting yard overnight
  - Midnight constraint: First departure after last arrival
- A shuffleboard shunting yard consists of Last-In-First-Out (LIFO) tracks connected in a tree-like configuration
- A conflict arises when a train is obstructed from departing
  - In LIFO tracks: earlier arriving train has to depart earlier
- Can often be resolved with expensive re-allocations
- Planning Domain Definition Language (PDDL) is used to model a planning problem

02 OBJECTIVE

Extend a PDDL model and optimize a planner to effectively avoid and resolve conflicts effectively in a shuffleboard shunting yard

03 METHODOLOGY

Portfolio planner:
- First planner does not allow re-allocations
- Second planner does allows for a single re-allocation per train

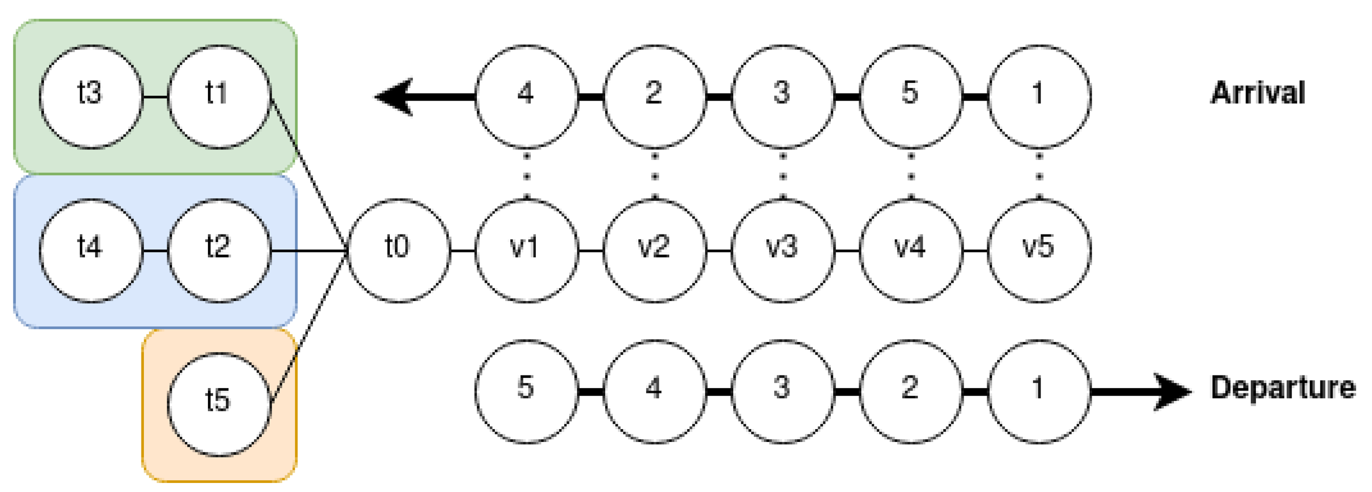
Should quickly find solution without re-allocations if one exists

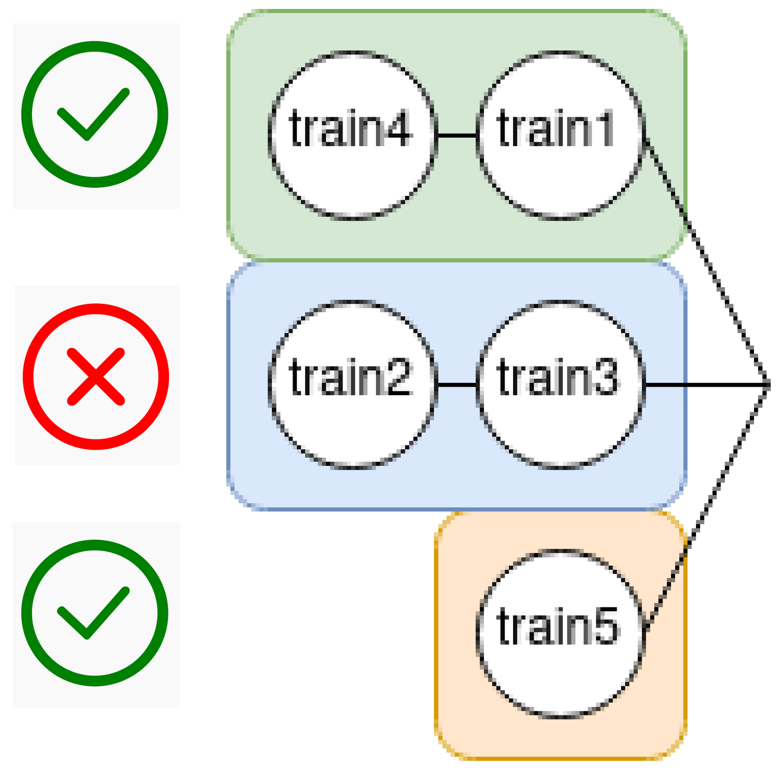
04 DOMAIN EXTENSIONS

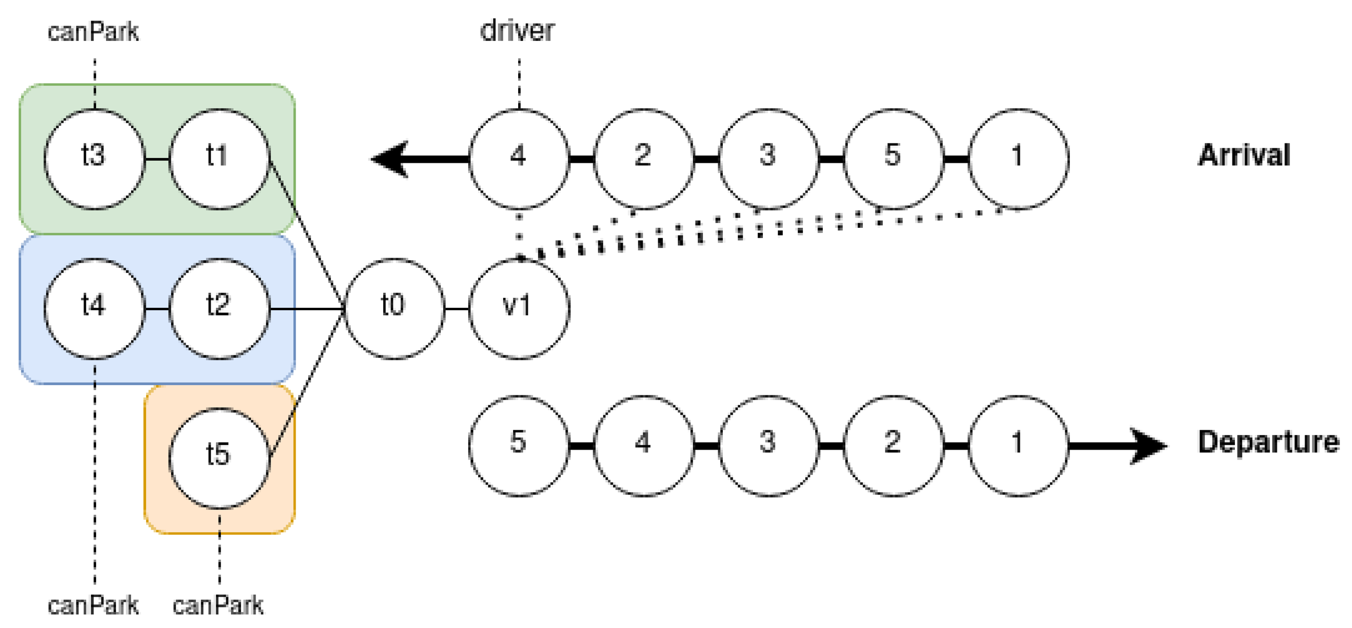
Both planners:
- Add direction
- Add driver
- Condense trainstation

Without re-allocation:
- Disallow trains from entering tracks if it creates a conflict

With re-allocations:
- Driver can switch to other trains
- Add re-allocation phases





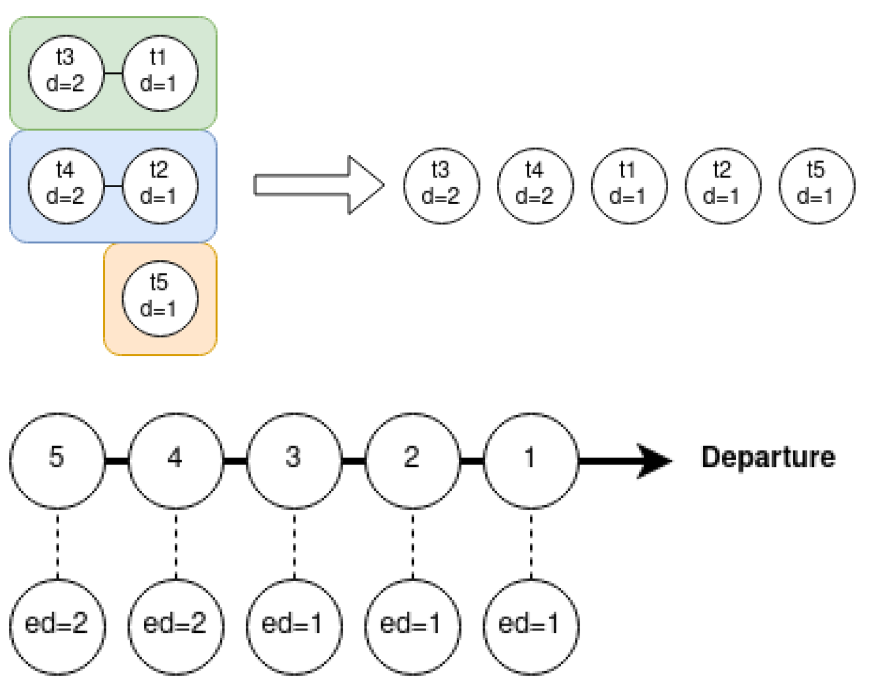


05 PLANNER OPTIMIZATION WITHOUT RE-ALLOCATION

ALLOCATION DURING PLANNING

Heuristic

- Balance progress against chance of conflict
- Progress = amount of trains parked
- Expected depth = map of trains in departing order to the depth of all track parts sorted on depth
- Chance of conflict is estimated by difference expected depth and actual depth

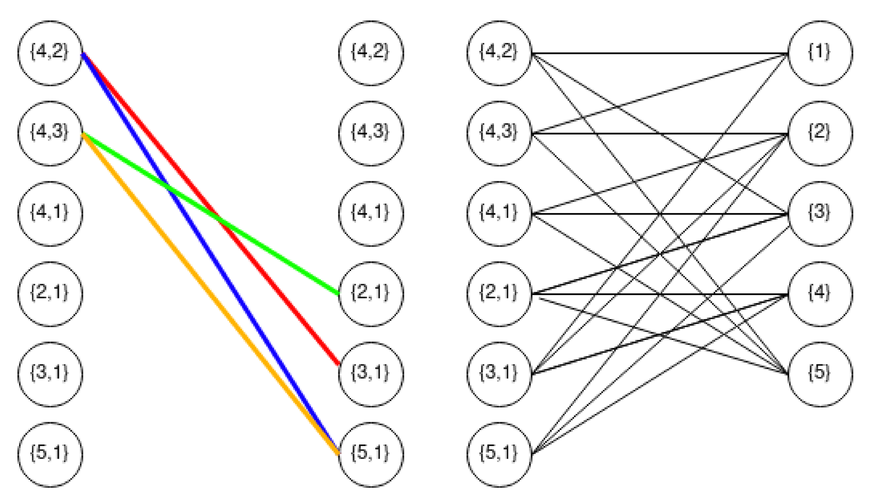


ALLOCATION WITH A PREPROCESSOR

First allocate the trains to tracks, then look for a plan which accommodates this allocation

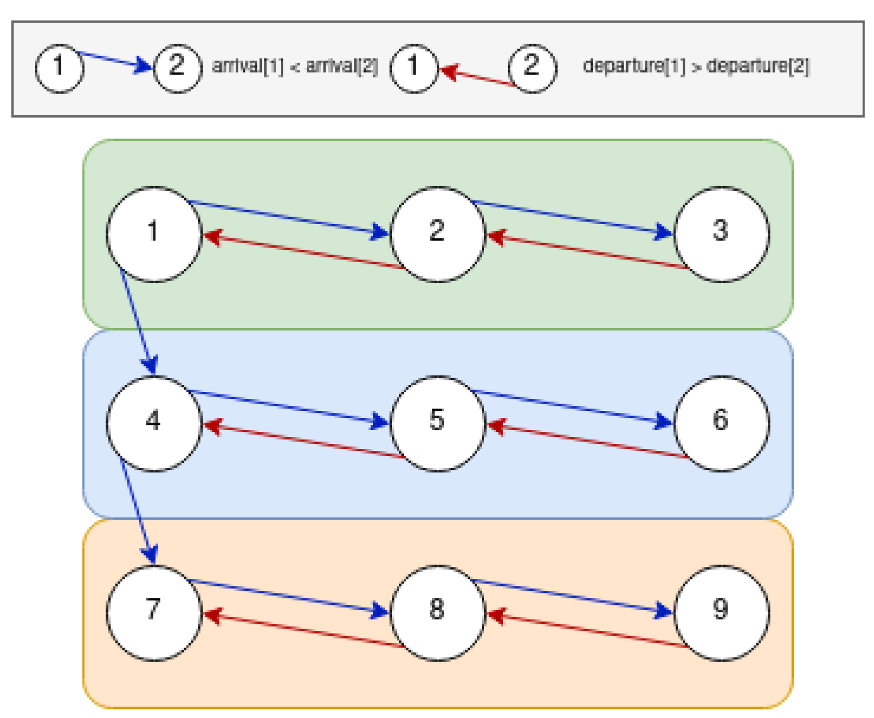
Set partitioning

- Generate all possible assignments for each track
- Pick a combination of track assignments covering every train
  - Start with the tracks which are most likely to cause a conflict
    - Compare connectivity between tracks



Constraint programming

- Define an array
  - each cell represents a track part
  - assigned value represents train
- Add an all different constraint over all cells
- For each connected track-part in the same track:
  - Add a constraint for arrival order
  - Add a constraint for departure order
- Add constraints between tracks of equal size to reduce symmetries



06 RESULTS

- Domain extension without re-allocation improved execution time significantly
- Domain extension with re-allocation has not been tested
- An implementation of the heuristic showed a small improvements in small problems
- An implementation of the constraint programming approach performed well across problems of all sizes.

Trains	Baseline	New domain	New domain + CP
5	1238ms	4ms	201ms
15	DNF	204ms	215ms
25	DNF	57152ms	228ms
40	DNF	DNF	708ms
50	DNF	DNF	1064ms

Table 1: Execution time of planners

07 CONCLUSION AND FUTURE WORK

- Building a preprocessor using constraint programming worked well
- Provided model for re-allocations should be implemented and tested
  - Research into optimizing planner using this model can be performed
- Set partitioning can be formulated an SAT, MILP or CP problem
- PDDL is good for planning research, but can be limiting factor in specific problem research
- After finding an allocation, a path-finding algorithm can be used instead of the PDDL planner
- Model can be extended by including: train lengths, train types, other shunting yard layouts, mixed-traffic, servicing actions and temporarily blocked tracks.