# Mixed-direction train shunting with numerical planning

*Approach to support train departures at any time during the shunting plan.*

## 1. Background

- Train Unit Shunting Problem (TUSP) [1]
- Algorithmic support by planning systems
- Planning Domain Definition Language (PDDL)

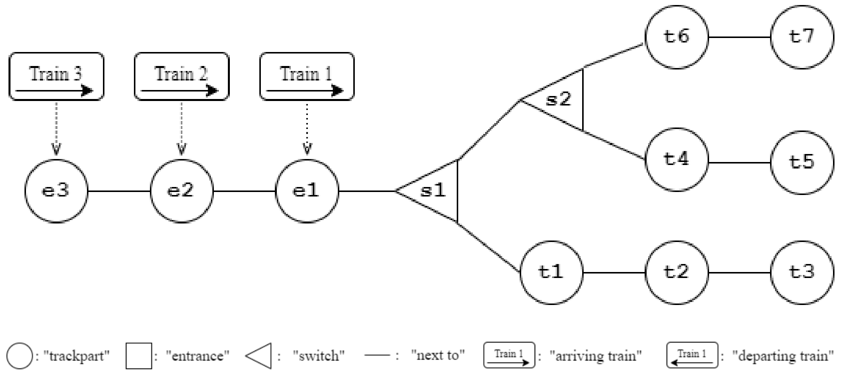


**Figure 1:** Schedule on the example domain. The initial order of the trains on the "arrival path" (e1, e2, e3) determines their arrival sequence. Similarly, their final order determines the departure schedule.

## 2. Problem Description

- Arrivals and departures modelled by an ordered list: *e1, e2, e3 ("arrival path").*
- *Free* predicate prevents trains to pass other units on a single track.
- No arrivals can happen after the first departure.
- *Can a planning system efficiently support mixed-directional train shunting in the planning domain?*

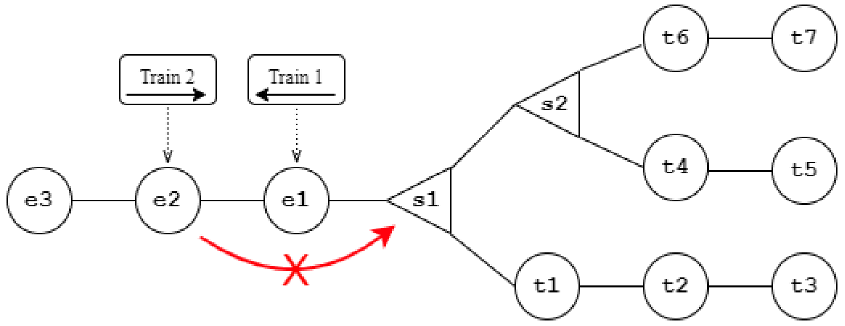


**Figure 2:** Illustration of the limiting factor in the example domain. If a train departs before all the other trains are in the yard, it blocks all future arrivals since arriving trains cannot "jump over" it.

## 3. Improved Domain

- The arriving and departing trains are held by a *set (e1),* thus allowing arrivals to happen after departures.
- *Numeric fluents* are used enforce the schedule.
- Each train has a unique arrival number. Departure times are given for train types.
- Switches can be omitted in the model since trains cannot be parked on them anyway.

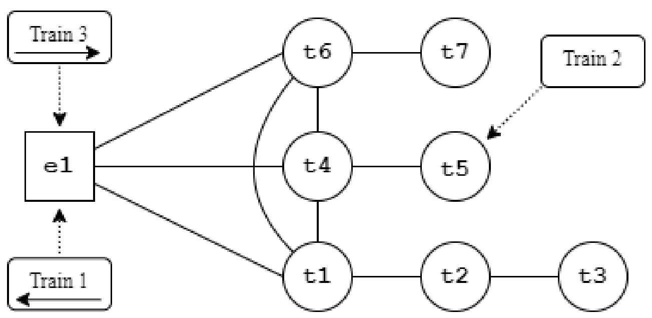


**Figure 3:** Problem model on the improved domain. All trains that are not in the yard are "held" by the entrance. Switches are omitted, tracks now connected directly.

| Unit | Type | Arrive | Depart |
|---|---|---|---|
| Train 1 | slt | 0 | 2* |
| Train 2 | slt | 1 | 4* |
| Train 3 | sng | 3 | 5 |

**Table 1:** Example definition of schedule in the new domain for the problem in Figure 3.

*: since the units with the same type can be used interchangeably, these two departures can be switched freely.

## 4. Experiment

| a) | Problem3 | | Problem5 | | Problem7 | | Problem8 | | Problem9 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost |
| EHC+BFS | 0.005s | 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| BFS | 0.006s | 0 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| BFS+H | 0.007s | 0 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| weighted A* | 0.006s | 0 | 0.076s | 6 | 1.881s | 4 | n/a | n/a | n/a | n/a |
| A* epsilon | 0.005s | 0 | 55.44s | 8 | 6.438s | 8 | 6.389s | 16 | n/a | n/a |
| EHC+A*eps | 0.006s | 4 | 57.628s* | 8 | 6.856s* | 8 | 9.354s* | 16 | n/a | n/a |

| b) | Problem7b | | Problem7c | | Problem8b | | Problem8c | | Problem8d | | Problem9b | | Problem9c | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost |
| weighted A* | 0.563s | 4 | 0.102s | 16 | 85.757s | 10 | 23.990s | 12 | 1581s | 6 | n/a | n/a | n/a | n/a |
| A* epsilon | 1.994s | 19 | 0.818s | 16 | 19.489s | 17 | 11.629s | 15 | 7.151s | 14 | n/a | n/a | n/a | n/a |

**Table 2:** Results of the first **(a)** and second **(b)** round of the experiment. N/a shows where the 30-minute timeout was reached before the algorithm could finish. ∗ shows where the EHC failed, and the alternative search method was applied.

- Used the MetricFF [2] planning system for the evaluation.
- Measured search speed and resulting plan cost.
- Defined the difficulty of a problem based on the number of trains.
- First round, all search methods were tested on problems with increasing difficulty.
- Second round further evaluated the best performing search methods on different problems.

## 5. Discussion on the Results

- Most cases EHC only find local maximum, not the goal. BFS uses zero node weights, thus it has to travers more of the relaxation graph.
- A* chooses the node with the least cost as next, therefore it mostly finds the cheapest solution. The epsilon variant also considers suboptimal nodes, thus usually finds solution faster.
- Suboptimal nodes lead to less cost-efficient solution.

## 6. Conclusions

- The worst-case time complexity of A* is $O(b^d)$, where $b$ is average successor nodes and $d$ is the length of the solution.
- Execution time grows exponentially with respect to solution length, which is not known prior to execution, and it increases with number of trains.
- *Mixed-direction shunting cannot be efficiently supported with the MetricFF planning system* due to its issues with scalability.


[1] R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman, "Shunting of passenger train units in a railway station," (in English), *Transp. Sci.,* Article vol. 39, no. 2, pp. 261-272, May 2005, doi: 10.1287/trsc.1030.0076.

[2] J. Hoffmann, "The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables," (in English), *J. Artif. Intell. Res.,* Article vol. 20, pp. 291-341, 2003, doi: 10.1613/jair.1144.



Author: *Ákos S. Sárkány*  Contact: *sarkany@student.tudelft.nl*  Supervisor: *Issa K. Hanou*  Responsible professor: *Sebastijan Dumančić*