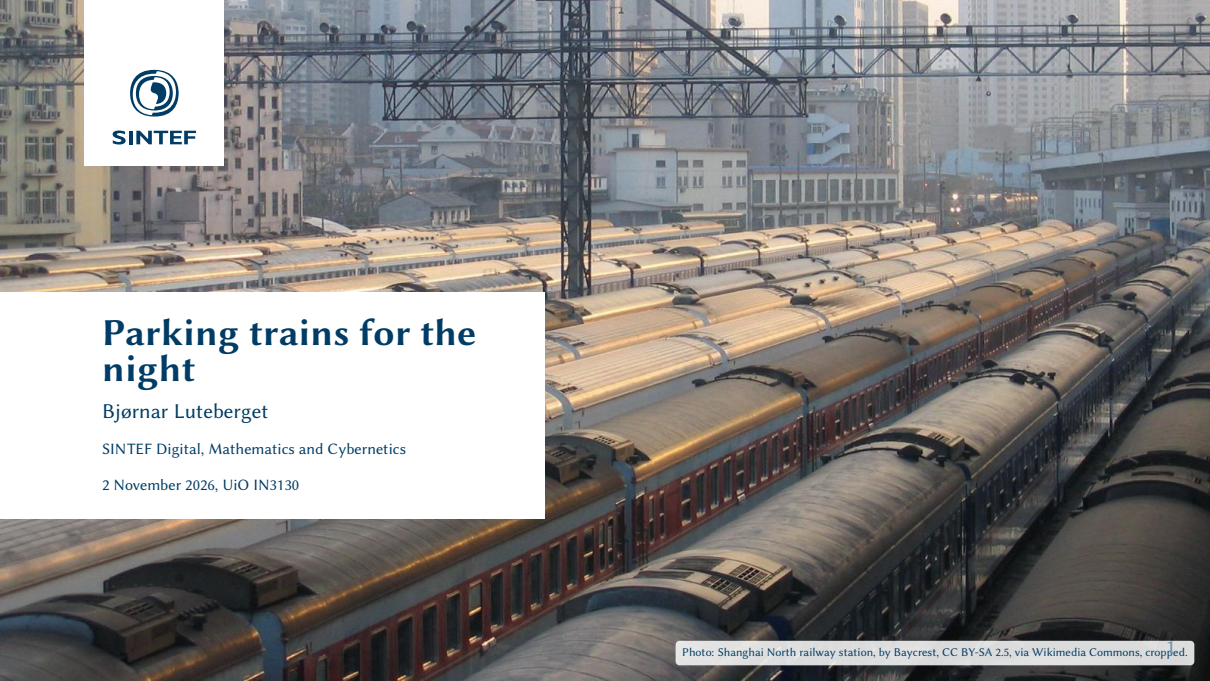


Parking trains for the night

Bjørnar Luteberget

SINTEF Digital, Mathematics and Cybernetics

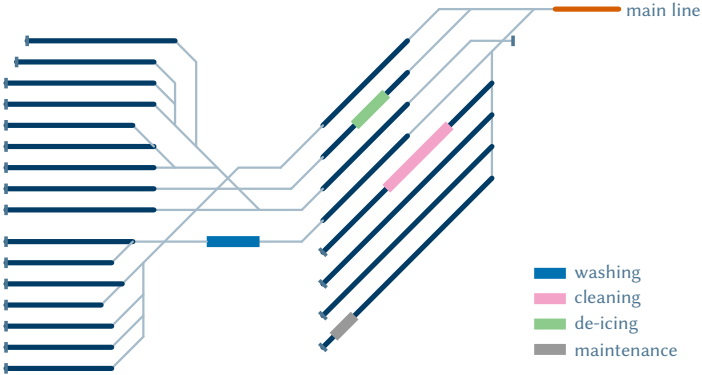
2 November 2026, UiO IN3130

Photo: Shanghai North railway station, by Baycrest, CC BY-SA 2.5, via Wikimedia Commons, cropped.

PART 1

The train depot stabling problem

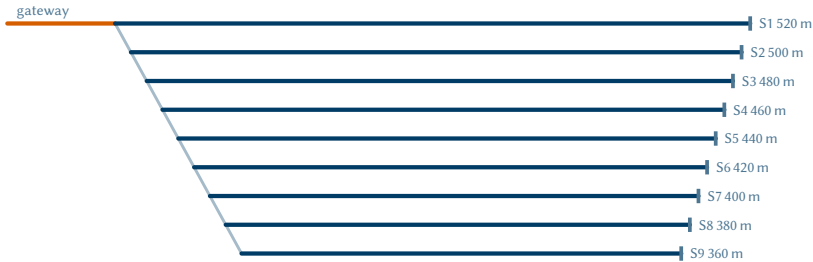
A depot: Cartesiusweg, Utrecht



- 24 parking tracks, 20 of them **dead ends**.
- They fan out from ladders on one side, so every train enters and leaves the same way.
- Trains come in for the night, park, and leave in the morning.
- Which train goes on which track?

A simplified version

Nine dead-end tracks, 3960 m in total.



- Every track is a **dead end**. Trains enter and leave at the west end, through the gateway.
- A track holds about **four trains**, depending on train lengths.
- Every track can be reached directly from the gateway. Only trains on the *same* track get in each other's way.

A track is a stack

Dead-end track: a **stack** (last in, first out)

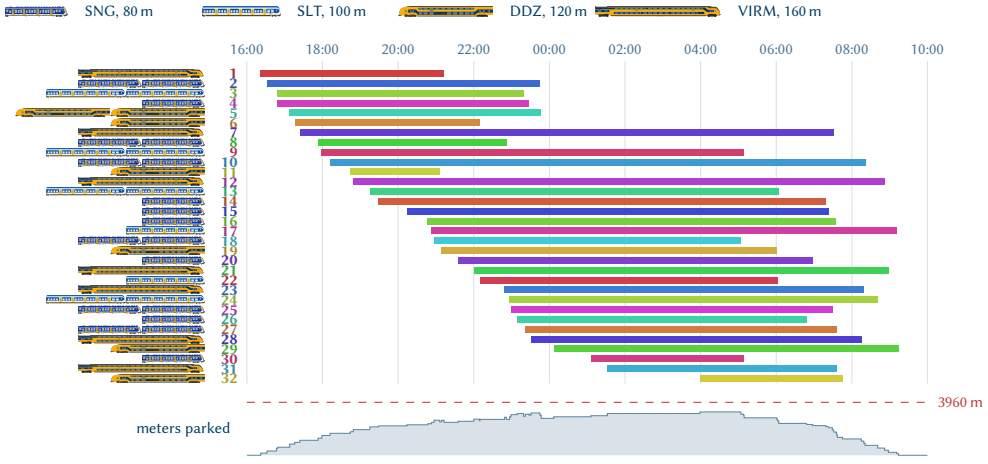


A cannot leave before B and C

Three trains arrive in the order A, B, C and stand on one track together. Of the six orders they could leave in, a dead end allows **exactly one**, namely C, B, A.

- A train that fits can still be **trapped** behind another.
- Lengths are in meters. Trains are 80 to 240 m long, tracks 360 to 520 m.
- A parked train never moves. It parks against the trains already there, and those stay until it has left.

One night: is there a plan?



The parking problem

Arrival and departure times are given. We only decide where each train stands.

Given. Trains t with arrival a_t, departure d_t and length ℓ_t. Dead-end tracks τ with length L_τ.

Decide. Where each train parks: a track, and a spot on it that the train keeps for its whole stay.

Valid when

- parked trains never overlap, and
- a train drives in and out at the open end, never past another parked train.

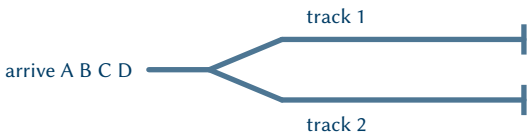
A simplified problem, but:

Even this is NP-hard. It contains coloring of circle graphs and bin packing.

R. Freling, R. M. Lentink, L. G. Kroon, D. Huisman. Shunting of passenger train units in a railway station. *Transportation Science* 39(2), 2005.

A puzzle

Four trains arrive in the order A, B, C, D.
Then they leave in the order B, D, A, C.



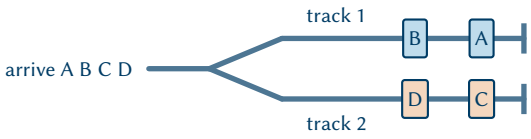
Two dead-end tracks, each long enough for two trains.

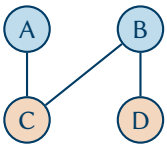
Rule. Two trains in the depot at the same time can share a dead-end track only if the *later arrival leaves first.*

T. Winter, U. T. Zimmermann. Real-time dispatch of trams in storage yards. *Annals of Operations Research* 96, 2000.

A puzzle

Four trains arrive in the order A, B, C, D.
Then they leave in the order B, D, A, C.





conflict graph
an edge when two trains
cannot share a track

Two dead-end tracks, each long enough for two trains.

Rule. Two trains in the depot at the same time can share a dead-end track only if the *later arrival leaves first*.

Pairs that cross. A–C (C arrives after A but leaves after A), B–C, B–D. So $\{A, B\}$ on one track and $\{C, D\}$ on the other. It is the only answer.

In general, track assignment is **graph coloring** of the conflict graph, one color per track, with lengths on top.

T. Winter, U. T. Zimmermann. Real-time dispatch of trams in storage yards. *Annals of Operations Research* 96, 2000.

PART 2

Problem solving with SAT

Boolean satisfiability (SAT)

A formula in **conjunctive normal form**:
$\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3)$

- **Literal**: a variable or its negation. **Clause**: an OR of literals. The formula: an AND of clauses.
- **SAT**: is there an assignment making every clause true? The assignment is a short **certificate**, so SAT is in NP.
- 2-SAT is polynomial (implication graph). 3-SAT is NP-complete.
- **Cook–Levin**: every problem in NP reduces to SAT.

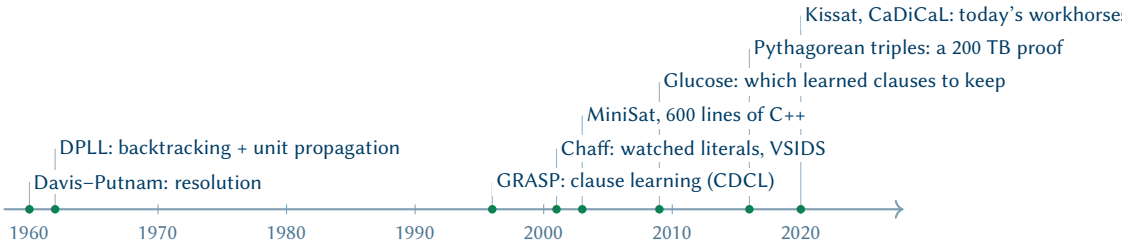
If we find a compact reduction from our problem into SAT, we can use a SAT solver to solve a SAT problem instead of our original problem.

This pays off because SAT solvers have become very fast.

Modern solvers backtrack as in lecture 8, with two additions. **Unit propagation**: a clause with all literals false but one forces that one. **Clause learning**: every conflict adds a new clause, so the solver never fails the same way twice.

J. P. Marques-Silva, K. A. Sakallah. GRASP: a search algorithm for propositional satisfiability. *IEEE Transactions on Computers* 48(5), 1999.

Sixty years of SAT solvers



- Formulas with **millions** of variables and clauses are routine. Solvers compete in a SAT Competition every year.
- Used for hardware and software verification, planning, cryptanalysis, proofs in mathematics.
- **SMT** (satisfiability modulo theories): SAT plus integers, arrays, bit-vectors. Z3, cvc5. Behind program verifiers.
- **CP-SAT** (Google OR-Tools): a SAT core with lazy clause generation, for scheduling.

Incremental SAT, assumptions and cores

The solver is a library object we keep alive and talk to.

```
void add_clause(Solver *S, int *lits, int n);  // any time, also between solves

int  solve     (Solver *S);                    // SAT or UNSAT
bool value     (Solver *S, int lit);           // after SAT: the assignment

// and no remove_clause
```

add_clause between solves keeps everything learned, since a learned clause still follows from the old clauses plus the new.

No remove_clause. A learned clause may depend on the removed one.

N. Eén, N. Sörensson. An extensible SAT-solver. *Theory and Applications of Satisfiability Testing (SAT 2003)*, LNCS 2919.

Incremental SAT, assumptions and cores

The solver is a library object we keep alive and talk to.

```
void add_clause(Solver *S, int *lits, int n);  // any time, also between solves
void assume    (Solver *S, int lit);           // holds for the next solve only
int  solve     (Solver *S);                    // SAT or UNSAT
bool value     (Solver *S, int lit);           // after SAT: the assignment
bool failed    (Solver *S, int lit);           // after UNSAT: is it in the core?
// and no remove_clause
```

add_clause between solves keeps everything learned, since a learned clause still follows from the old clauses plus the new.

No remove_clause. A learned clause may depend on the removed one.

assume sets a literal for one call. Assumptions are decisions, not clauses.

failed after UNSAT names the assumptions behind it, an **UNSAT core**. In our problem, the core is a set of trains that cannot all be parked. Drop one and solve again to shrink it.

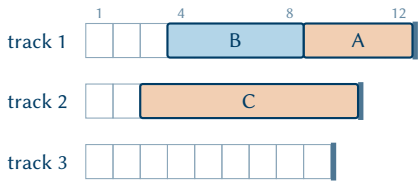
N. Eén, N. Sörensson. An extensible SAT-solver. *Theory and Applications of Satisfiability Testing (SAT 2003)*, LNCS 2919.

PART 3

The parking problem as a formula

Model 1: explicit positions

Cut every track into 20 m cells, numbered from the west end. Each train covers a run of cells.



$$x_{A,1,9} = x_{B,1,4} = x_{C,2,3} = 1$$

A (80 m) arrived first and stands deepest.
B (100 m) came later and stands west of it.

Idea. Decide for every train the track *and the exact cell* where its west end stands.

Then the rules are easy to state.

- Every train stands somewhere.
- If a newer train leaves before an older one on the same track, it stands west of it.
- Two trains that cross never share a track.

Capacity and fixed spots come for free. With equal lengths, the cells become *slots*.

Model 1: variables and clauses

Given. Trains $t \in T$ with arrival a_t, departure d_t and length c_t cells. Tracks $\tau \in K$ with length C_τ cells.

Variables. For each $t \in T$, $\tau \in K$ and $p \in P_{t,\tau} = \{1, \ldots, C_\tau - c_t + 1\}$:

$$x_{t,\tau,p} \in \{0, 1\} \qquad \text{(train } t \text{ stands on track } \tau \text{ with its west end in cell } p\text{)}$$

Clauses.

Every train parks. For each $t \in T$:

$$\bigvee_{\tau \in K} \bigvee_{p \in P_{t,\tau}} x_{t,\tau,p}$$

Two trains on one track. For each $i, j \in T$ where i arrives before j, each $\tau \in K$, $p \in P_{j,\tau}$, $q \in P_{i,\tau}$:



cross ($a_j < d_i < d_j$)
i cannot get out past j.

$$\neg x_{j,\tau,p} \vee \neg x_{i,\tau,q}$$
for every p, q



nested ($d_j < d_i$)
j must stand west of i.

$$\neg x_{j,\tau,p} \vee \neg x_{i,\tau,q}$$
when $p + c_j > q$



apart ($d_i < a_j$)
Never there together.

no clause

What is wrong with the positions model

- **Symmetry.** Only the track and the order on it matter, not the exact cell. A train with room to spare can stand in many cells, and every combination is a different but equivalent assignment. The solver explores them all when it fails.
- **Size.** For n trains, m tracks and c cells per track, about $n\,m\,c$ variables and $n^2\,m\,c^2$ clauses. On our night 4608 variables and 0.9 million clauses, and finer cells make it worse.
- **UNSAT is slow.** When the trains do not fit, the solver can only find out by trying placements, one after another. On our night it gives no answer in 15 minutes.

How the problem is encoded decides what the solver can prove fast. With positions, it has to rule out placements one by one.

Only the track matters

On a dead end, the exact spot follows by itself.

A set of trains can share a dead-end track exactly when

(LIFO) no two of them **cross**: never $a_i < a_j < d_i < d_j$, and

(CAP) at every moment, the trains present fit in L_τ.



a crossing pair

Why. Let each arriving train park against the trains already there. By (CAP) it fits. Those trains are older, and by (LIFO) they stay until it has left. So no train is ever trapped, and no train ever has to move.

So we do not need to decide spots at all, only a **track** for each train.

Model 2: track variables

$y_{t,\tau}$: train t stands on track τ.

- **Every train somewhere.** $\bigvee_\tau y_{t,\tau}$.
- **No crossing.** For every crossing pair i, j and every track,



a crossing pair

$$\neg y_{i,\tau} \vee \neg y_{j,\tau}.$$

- **Capacity.** For each track τ and each arrival time q, with T_q the trains present at q,

$$\sum_{t \in T_q} \ell_t \, y_{t,\tau} \;\leq\; L_\tau.$$

Far fewer variables and constraints than with positions: on our night **288 variables** instead of 4608.

To find which trains block a plan, train t gets the clause $\neg s_t \vee \bigvee_\tau y_{t,\tau}$, and we assume every s_t.

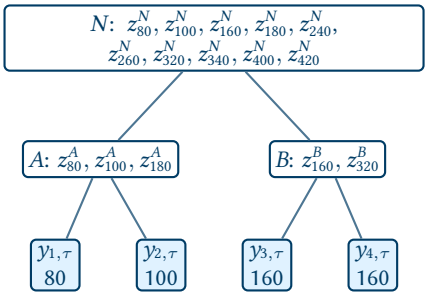
But a solver only understands clauses.

The naive way: forbid every set of trains that is too long together. Exponentially many clauses.

The trick: auxiliary variables that **add up the lengths**.

The generalized totalizer

Encodes $\sum_{t \in T_q} \ell_t \, y_{t,\tau} \leq L_\tau$ for one track τ and one arrival time q.



$L_\tau = 420$: $180 + 320 = 500$ is never named

A binary tree over the trains present at q.

Leaves. One per train, its variable $y_{t,\tau}$ and its length ℓ_t.

Inner nodes. One new variable for each total that the trains below can reach, up to L_τ. z^A_{180} means "the trains below A on track τ total 180 m or more".

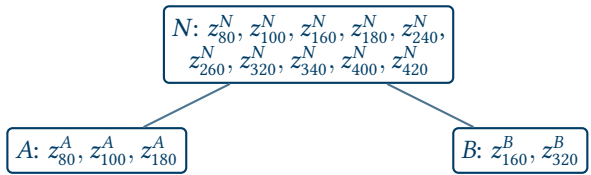
Too long. Totals above L_τ get no variable. Clauses forbid the combinations that would reach them.

Size grows with the number of *distinct* totals, not with meters. Few train lengths, few totals.

S. Joshi, R. Martins, V. Manquinho. Generalized totalizer encoding for pseudo-Boolean constraints. *Principles and Practice of Constraint Programming (CP 2015)*, LNCS 9255, 2015.

A totalizer tree node

Node N with children A and B, track length $L_\tau = 420$.



Variables. For each total $s \le L_\tau$ that A and B can reach together:

$$z^N_s \in \{0, 1\} \qquad \text{(the lengths below } N \text{ sum to } s \text{ or more)}$$

Clauses, for each total a of A and b of B:

Add up	if $a + b \le L_\tau$	$\neg z^A_a \vee \neg z^B_b \vee z^N_{a+b}$	e.g. $\neg z^A_{100} \vee \neg z^B_{320} \vee z^N_{420}$
Too long	if $a + b > L_\tau$	$\neg z^A_a \vee \neg z^B_b$	e.g. $\neg z^A_{180} \vee \neg z^B_{320}$

a and b may also be 0, meaning nothing below. That literal is left out, as in $\neg z^A_{80} \vee z^N_{80}$. If a child is a leaf, its only variable is the train's own $y_{t,\tau}$, with total ℓ_t.

Comparing the two models

Solving it on the night we looked at.

	Explicit positions	Track variables + totalizers
Clauses	≈900 000	68 000
Time to an answer	none in 15 minutes	5 s

The second formula leaves out what does not matter, the exact spot on the track. So it is much smaller, and the solver has far less to try.

PART 4

Solving our night

Our night has no plan

SAT says UNSAT after 5 seconds. Shrinking the core takes 5 seconds more. It names 10 trains.



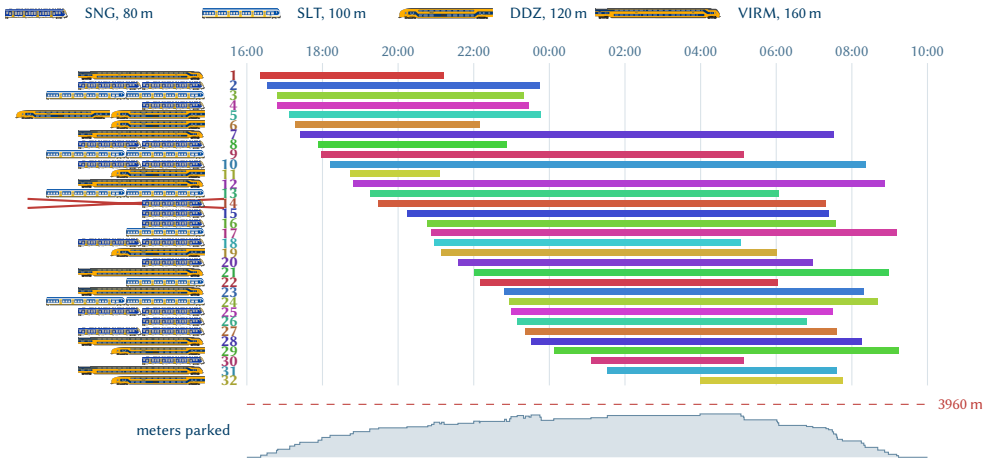
Trains 3, 4, 5, 9, 13, 14, 15, 16, 23 and 24 all stand in the depot from 22:57 to 23:19 (shaded).

They **leave in the order they arrived**. So every two of them cross, and no two can share a dead end.

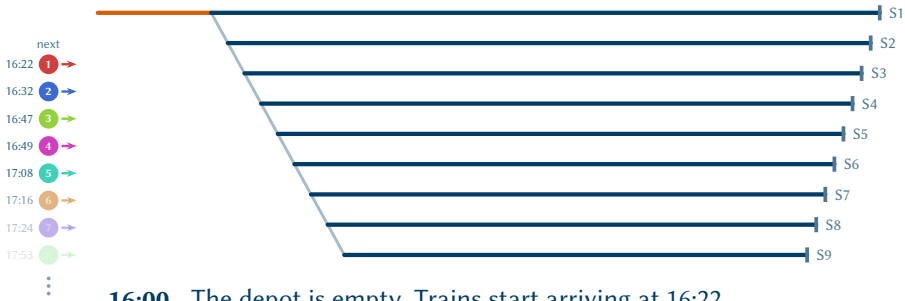
Ten trains, nine tracks. The depot is only 82 % full, but no plan exists.

Send train 14 somewhere else

Remove one train of the ten. Now SAT finds a plan in 0.06 seconds. With explicit positions it takes 100 seconds.

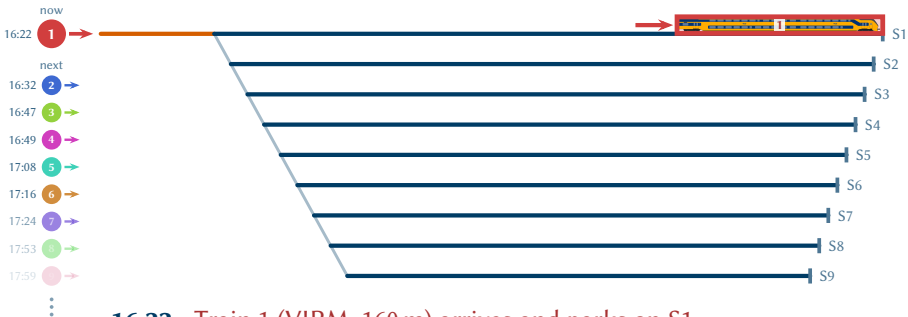


The plan, event by event



16:00 The depot is empty. Trains start arriving at 16:22.

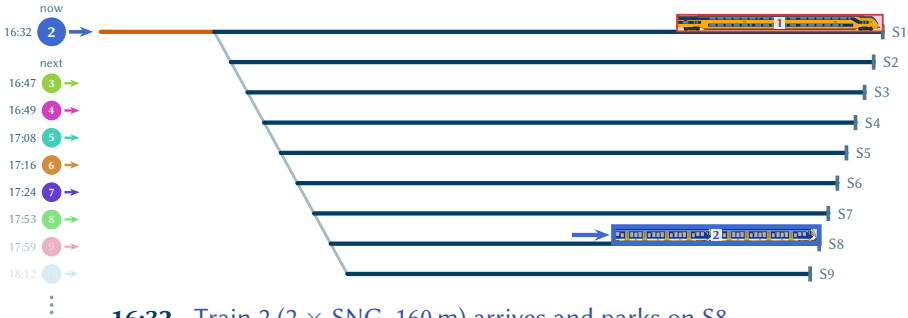
The plan, event by event



16:22 Train 1 (VIRM, 160 m) arrives and parks on S1.

1 train, 160 m parked event 1 of 62

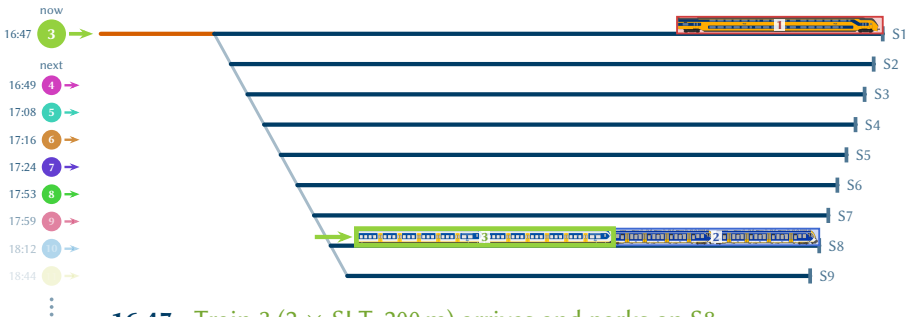
The plan, event by event



16:32 Train 2 (2 × SNG, 160 m) arrives and parks on S8.

2 trains, 320 m parked event 2 of 62

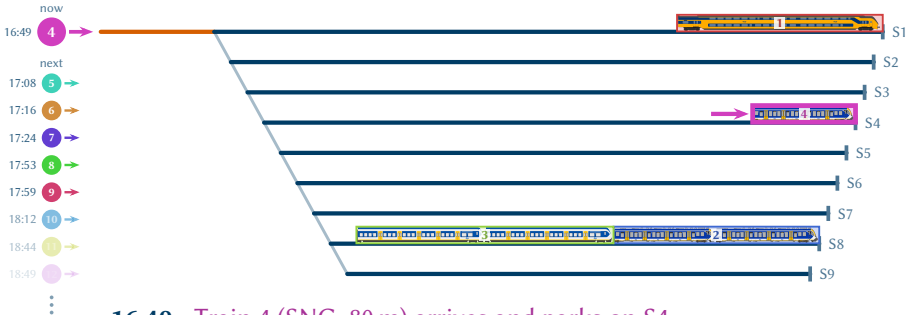
The plan, event by event



16:47 Train 3 (2 × SLT, 200 m) arrives and parks on S8.

3 trains, 520 m parked event 3 of 62

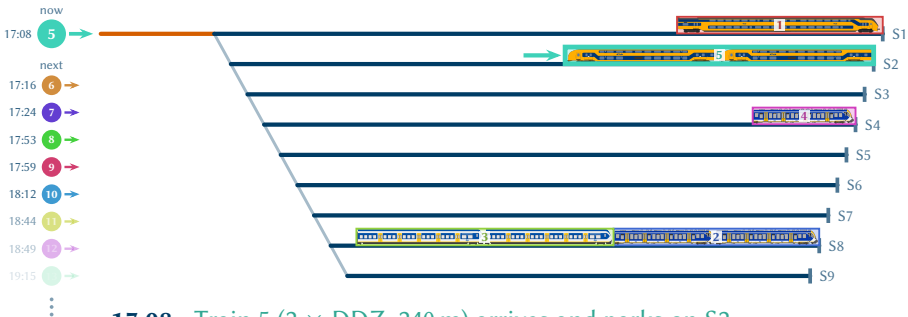
The plan, event by event



16:49 Train 4 (SNG, 80 m) arrives and parks on S4.

4 trains, 600 m parked event 4 of 62

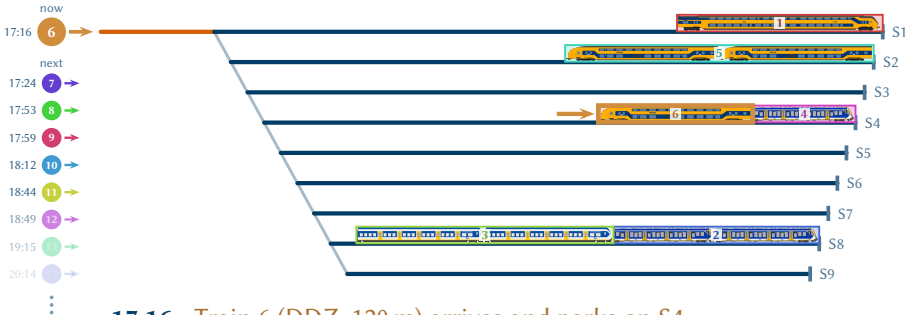
The plan, event by event



17:08 Train 5 (2 × DDZ, 240 m) arrives and parks on S2.

5 trains, 840 m parked event 5 of 62

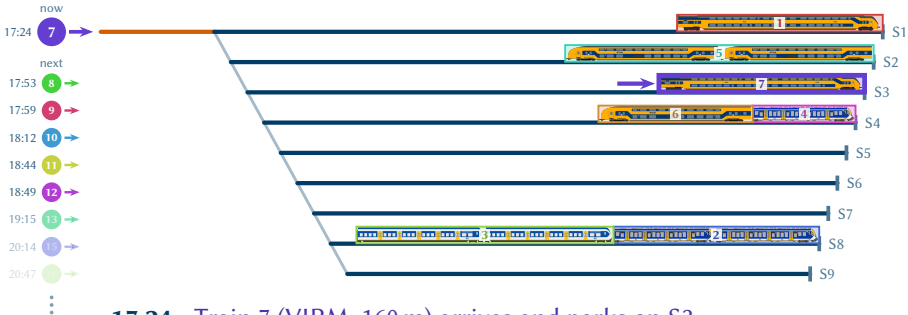
The plan, event by event



17:16 Train 6 (DDZ, 120 m) arrives and parks on S4.

6 trains, 960 m parked event 6 of 62

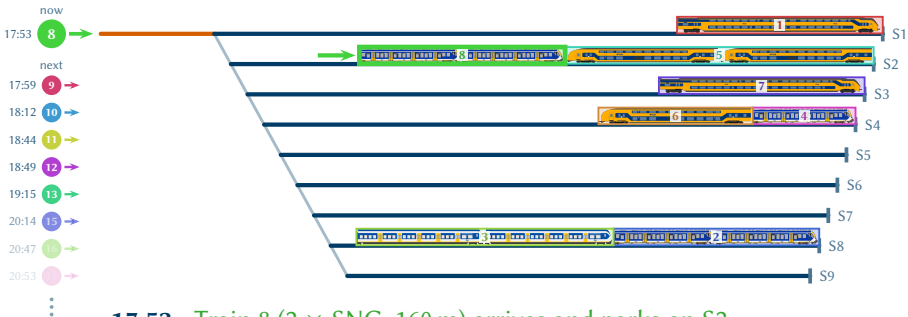
The plan, event by event



17:24 Train 7 (VIRM, 160 m) arrives and parks on S3.

7 trains, 1120 m parked event 7 of 62

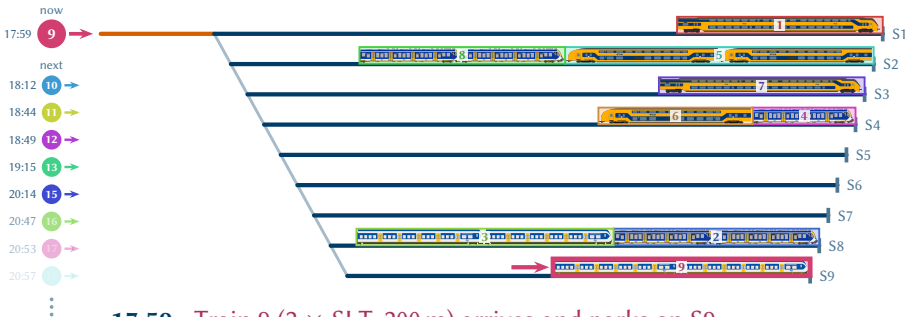
The plan, event by event



17:53 Train 8 (2 × SNG, 160 m) arrives and parks on S2.

8 trains, 1280 m parked event 8 of 62

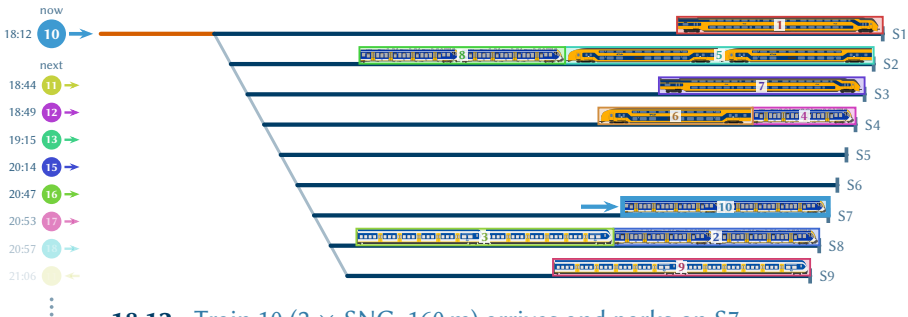
The plan, event by event



17:59 Train 9 (2 × SLT, 200 m) arrives and parks on S9.

9 trains, 1480 m parked event 9 of 62

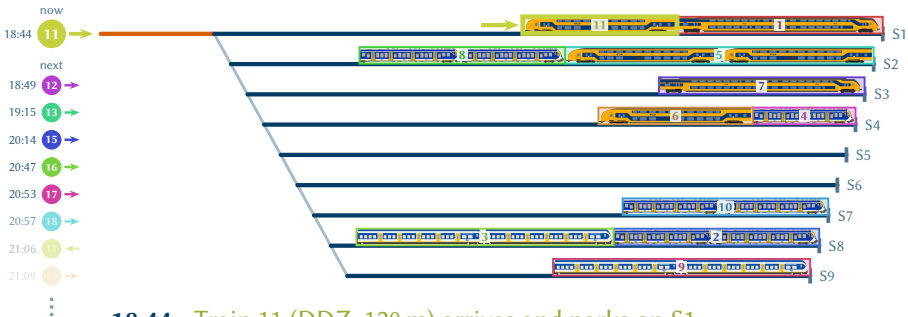
The plan, event by event



18:12 Train 10 (2 × SNG, 160 m) arrives and parks on S7.

10 trains, 1640 m parked event 10 of 62

The plan, event by event



18:44 Train 11 (DDZ, 120 m) arrives and parks on S1.

11 trains, 1760 m parked event 11 of 62

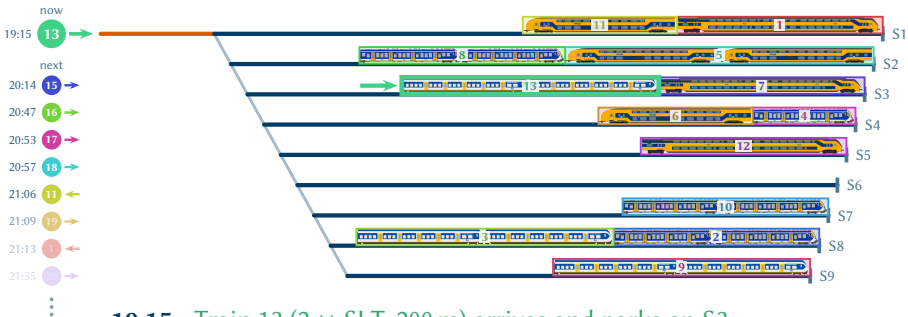
The plan, event by event



18:49 Train 12 (VIRM, 160 m) arrives and parks on S5.

12 trains, 1920 m parked event 12 of 62

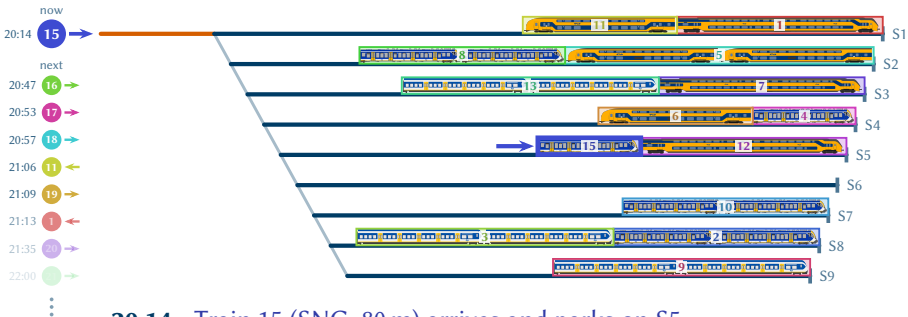
The plan, event by event



19:15 Train 13 (2 × SLT, 200 m) arrives and parks on S3.

13 trains, 2120 m parked event 13 of 62

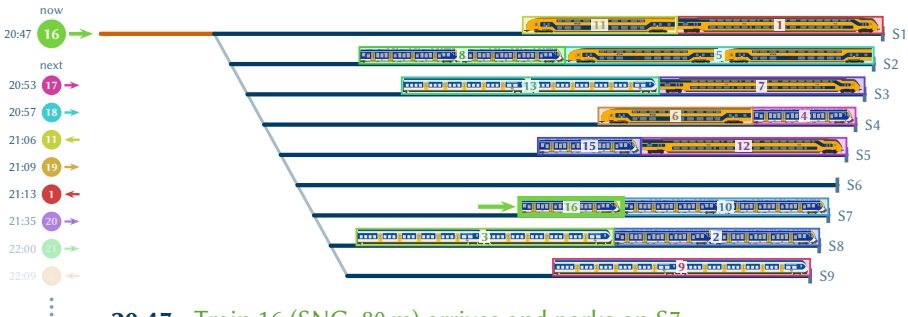
The plan, event by event



20:14 Train 15 (SNG, 80 m) arrives and parks on S5.

14 trains, 2200 m parked event 14 of 62

The plan, event by event



20:47 Train 16 (SNG, 80 m) arrives and parks on S7.

15 trains, 2280 m parked event 15 of 62

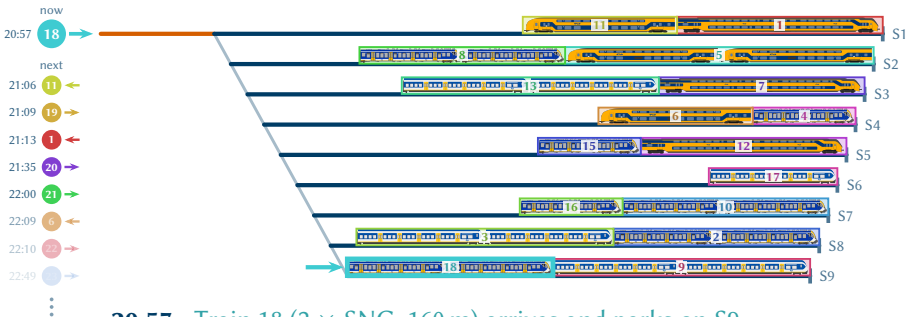
The plan, event by event



20:53 Train 17 (SLT, 100 m) arrives and parks on S6.

16 trains, 2380 m parked event 16 of 62

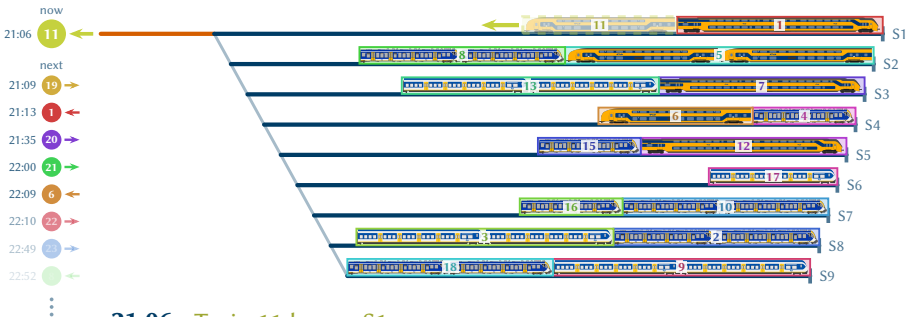
The plan, event by event



20:57 Train 18 (2 × SNG, 160 m) arrives and parks on S9.

17 trains, 2540 m parked event 17 of 62

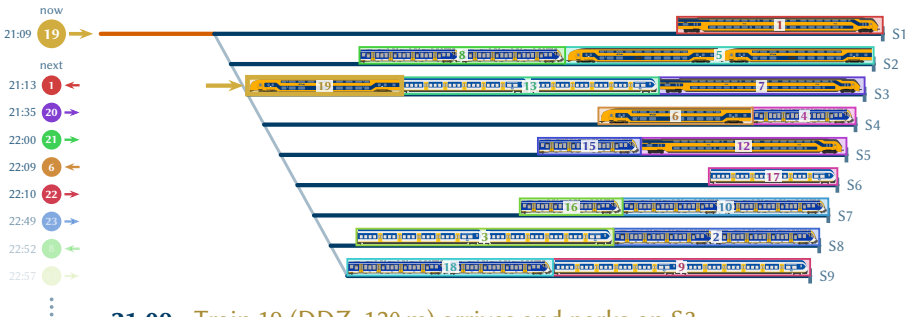
The plan, event by event



21:06 Train 11 leaves S1.

16 trains, 2420 m parked event 18 of 62

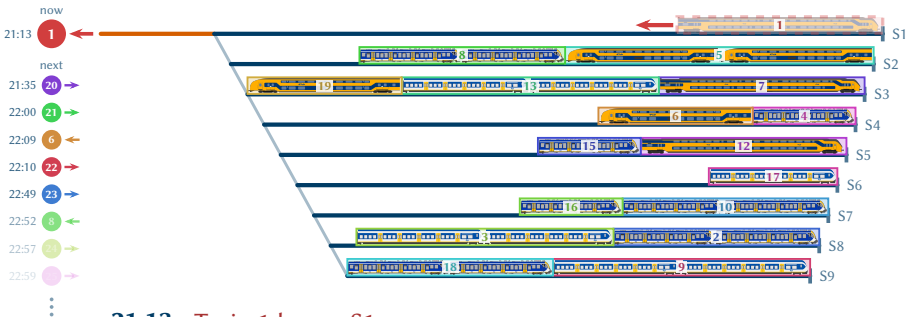
The plan, event by event



21:09 Train 19 (DDZ, 120 m) arrives and parks on S3.

17 trains, 2540 m parked event 19 of 62

The plan, event by event



21:13 Train 1 leaves S1.

16 trains, 2380 m parked event 20 of 62

The plan, event by event



21:35 Train 20 (SNG, 80 m) arrives and parks on S5.

17 trains, 2460 m parked event 21 of 62

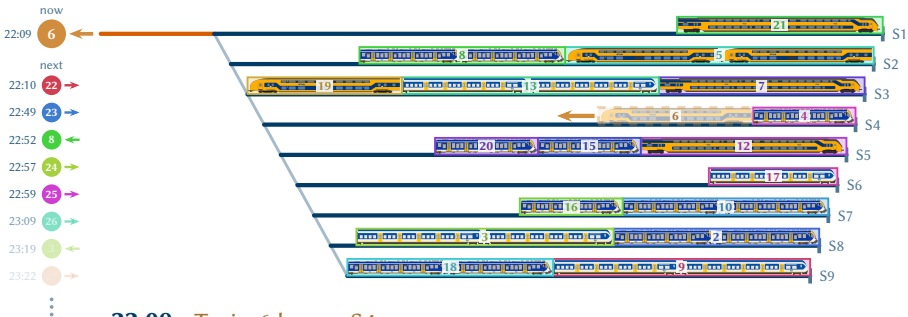
The plan, event by event



22:00 Train 21 (VIRM, 160 m) arrives and parks on S1.

18 trains, 2620 m parked event 22 of 62

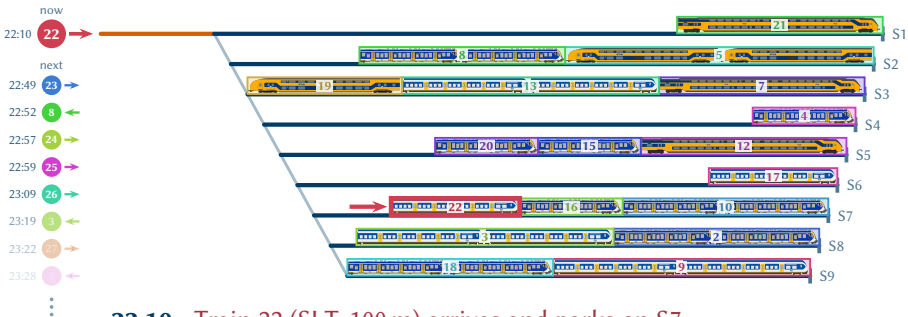
The plan, event by event



22:09 Train 6 leaves S4.

17 trains, 2500 m parked event 23 of 62

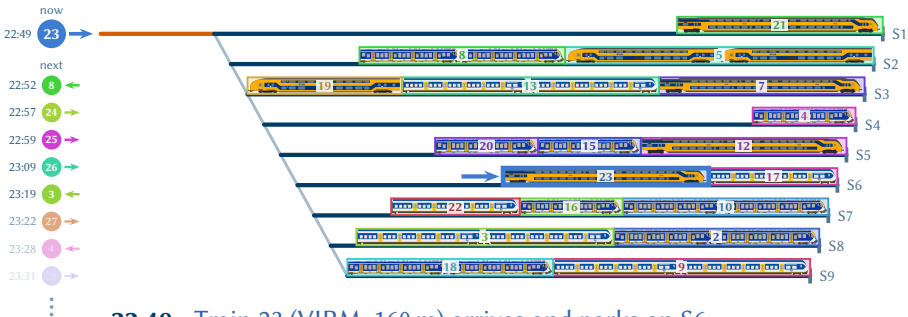
The plan, event by event



22:10 Train 22 (SLT, 100 m) arrives and parks on S7.

18 trains, 2600 m parked event 24 of 62

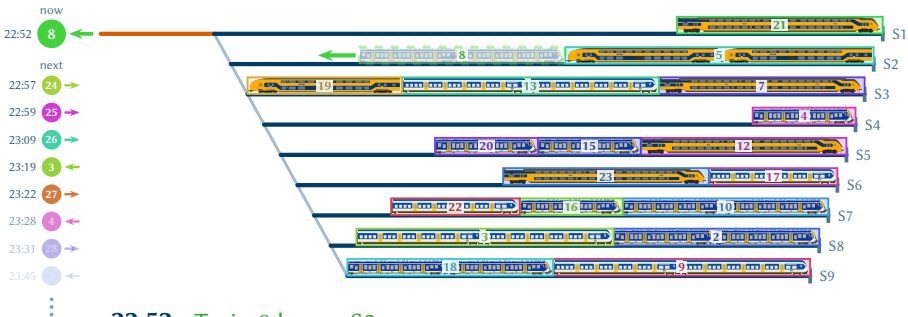
The plan, event by event



22:49 Train 23 (VIRM, 160 m) arrives and parks on S6.

19 trains, 2760 m parked event 25 of 62

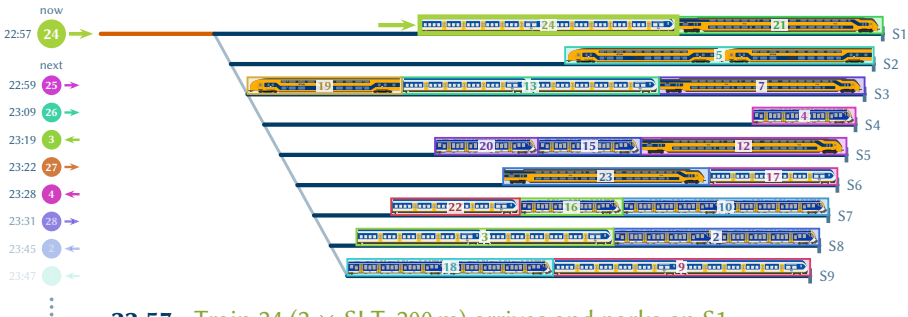
The plan, event by event



22:52 Train 8 leaves S2.

18 trains, 2600 m parked event 26 of 62

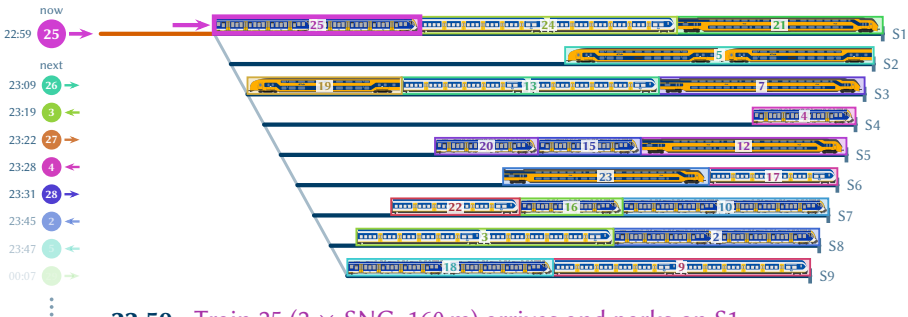
The plan, event by event



22:57 Train 24 (2 × SLT, 200 m) arrives and parks on S1.

19 trains, 2800 m parked event 27 of 62

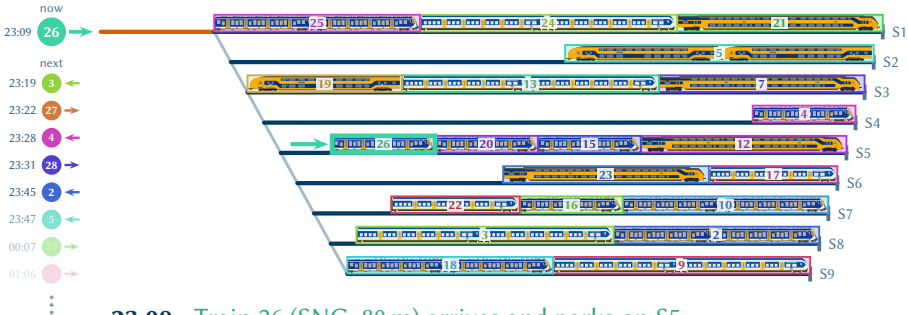
The plan, event by event



22:59 Train 25 (2 × SNG, 160 m) arrives and parks on S1.

20 trains, 2960 m parked event 28 of 62

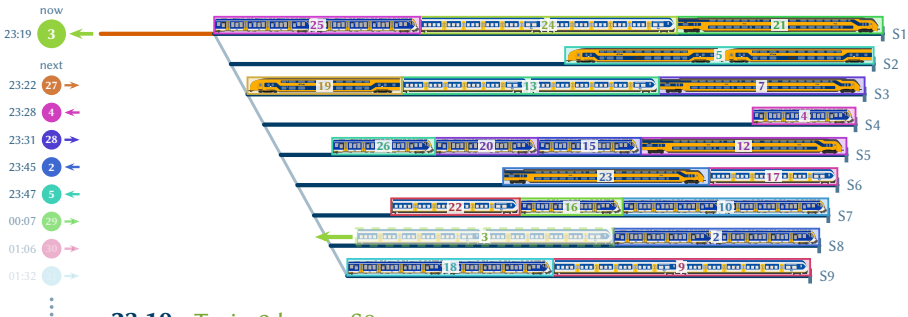
The plan, event by event



23:09 Train 26 (SNG, 80 m) arrives and parks on S5.

21 trains, 3040 m parked event 29 of 62

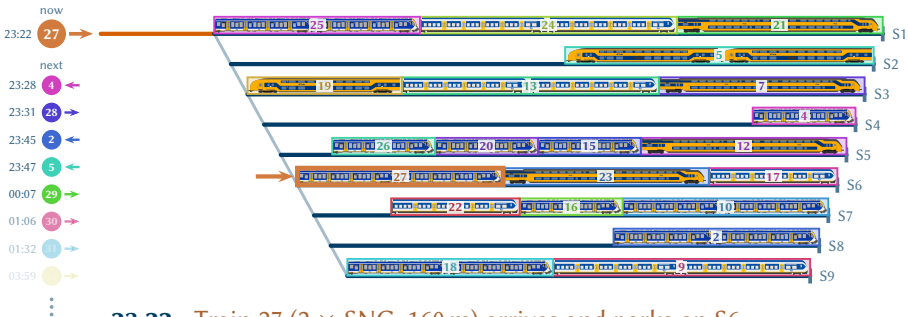
The plan, event by event



23:19 Train 3 leaves S8.

20 trains, 2840 m parked event 30 of 62

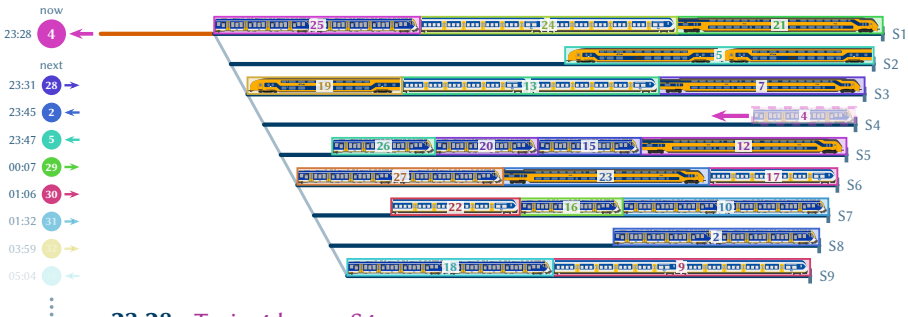
The plan, event by event



23:22 Train 27 (2 × SNG, 160 m) arrives and parks on S6.

21 trains, 3000 m parked event 31 of 62

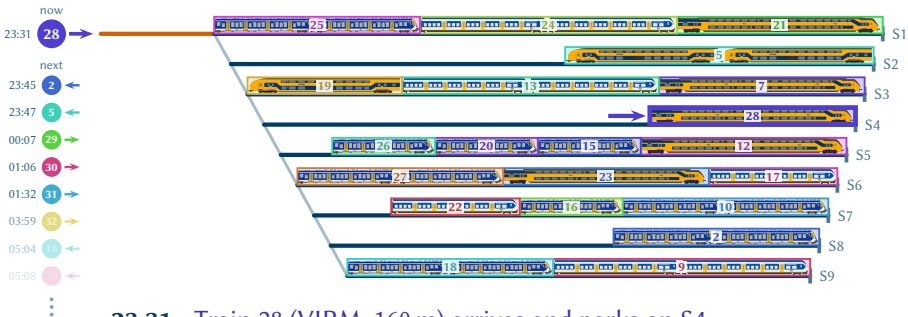
The plan, event by event



23:28 Train 4 leaves S4.

20 trains, 2920 m parked event 32 of 62

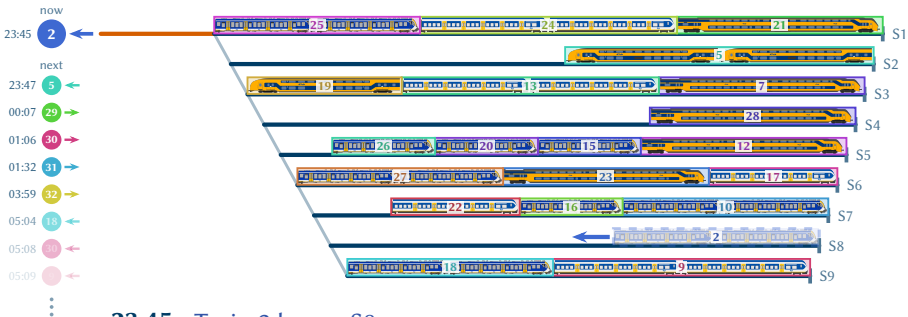
The plan, event by event



23:31 Train 28 (VIRM, 160 m) arrives and parks on S4.

21 trains, 3080 m parked event 33 of 62

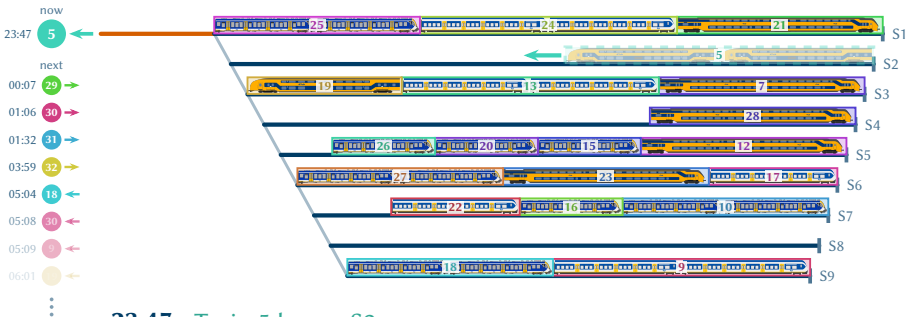
The plan, event by event



23:45 Train 2 leaves S8.

20 trains, 2920 m parked event 34 of 62

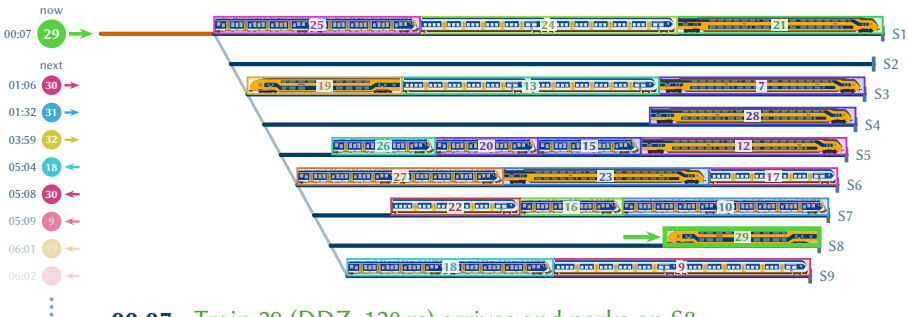
The plan, event by event



23:47 Train 5 leaves S2.

19 trains, 2680 m parked event 35 of 62

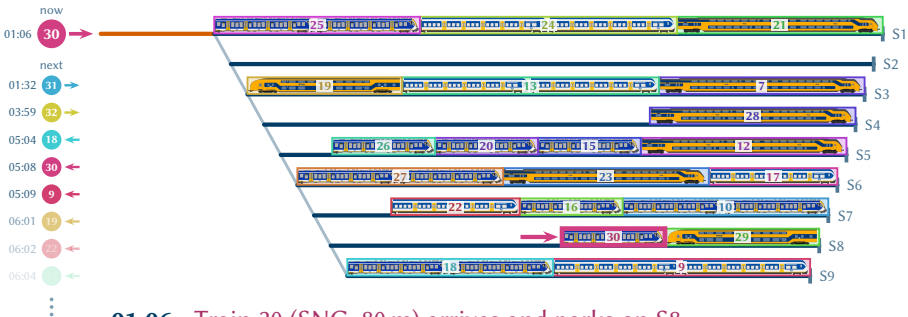
The plan, event by event



00:07 Train 29 (DDZ, 120 m) arrives and parks on S8.

20 trains, 2800 m parked event 36 of 62

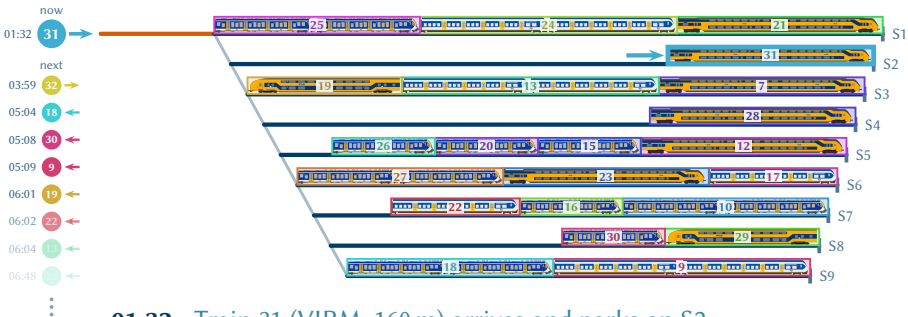
The plan, event by event



01:06 Train 30 (SNG, 80 m) arrives and parks on S8.

21 trains, 2880 m parked event 37 of 62

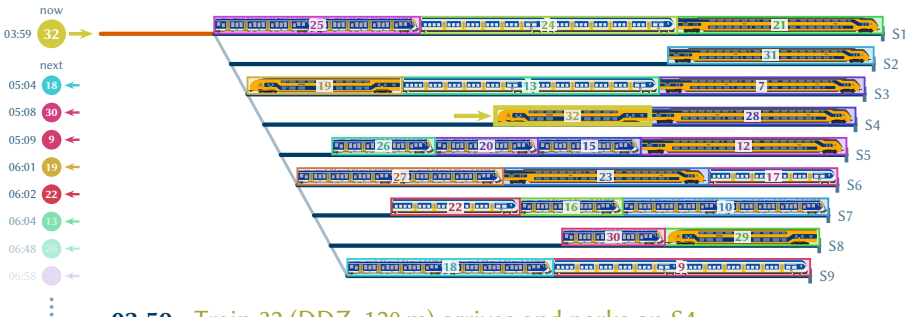
The plan, event by event



01:32 Train 31 (VIRM, 160 m) arrives and parks on S2.

22 trains, 3040 m parked event 38 of 62

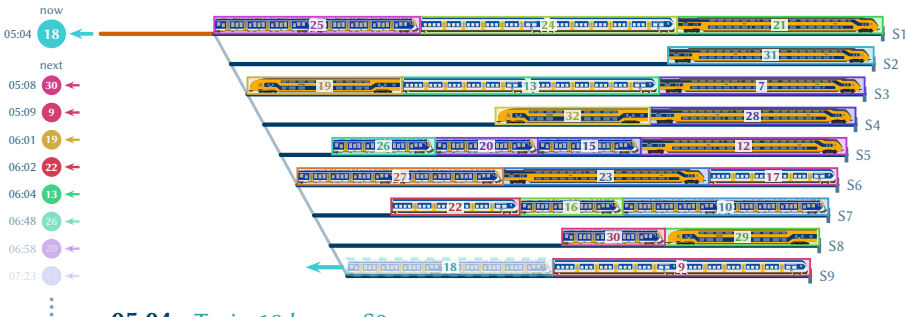
The plan, event by event



03:59 Train 32 (DDZ, 120 m) arrives and parks on S4.

23 trains, 3160 m parked event 39 of 62

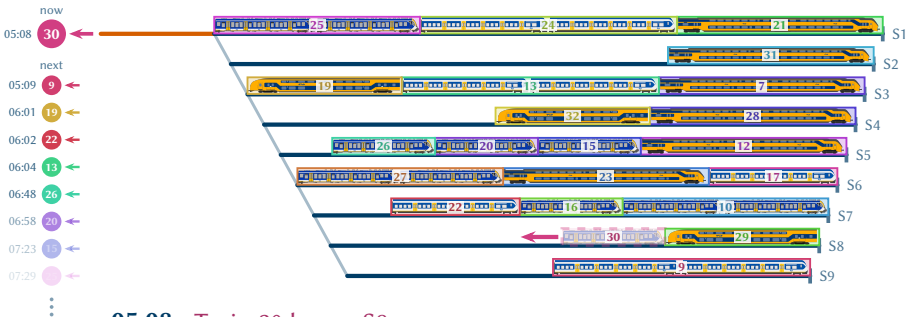
The plan, event by event



05:04 Train 18 leaves S9.

22 trains, 3000 m parked event 40 of 62

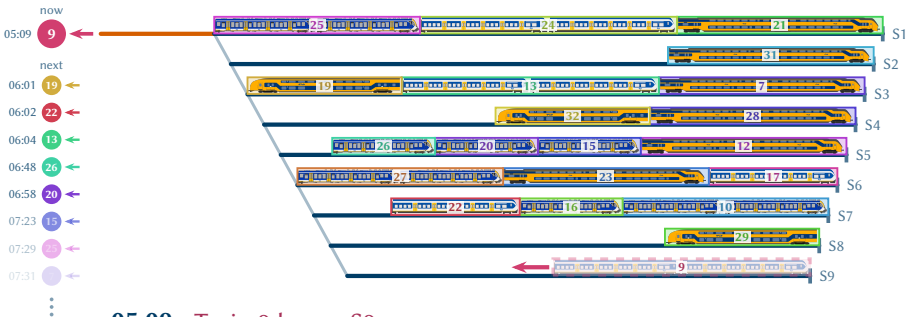
The plan, event by event



05:08 Train 30 leaves S8.

21 trains, 2920 m parked event 41 of 62

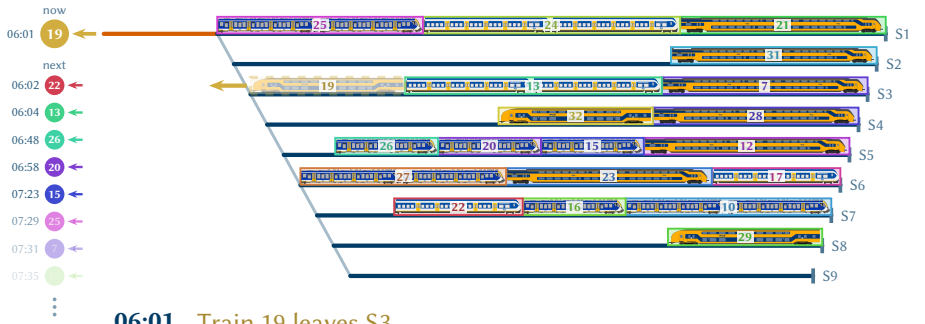
The plan, event by event



05:09 Train 9 leaves S9.

20 trains, 2720 m parked event 42 of 62

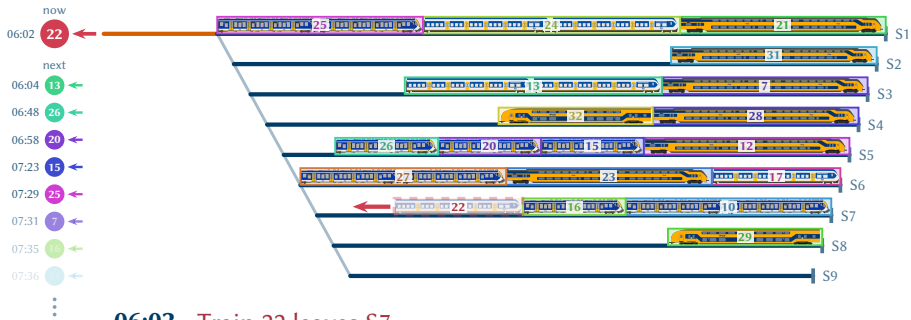
The plan, event by event



06:01 Train 19 leaves S3.

19 trains, 2600 m parked event 43 of 62

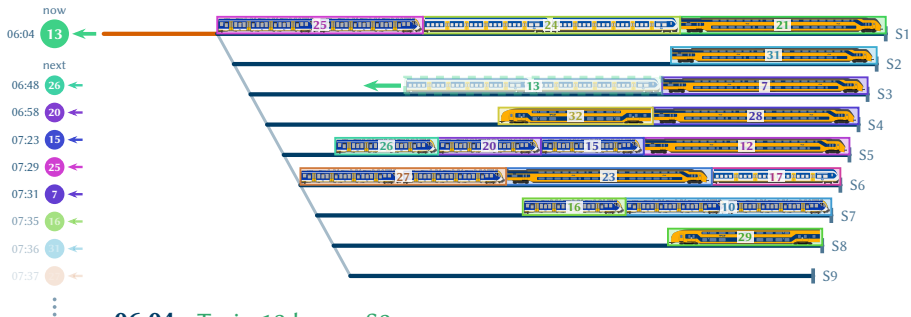
The plan, event by event



06:02 Train 22 leaves S7.

18 trains, 2500 m parked event 44 of 62

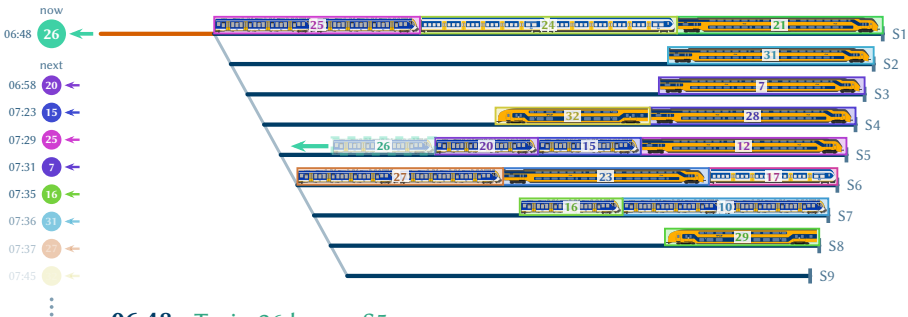
The plan, event by event



06:04 Train 13 leaves S3.

17 trains, 2300 m parked event 45 of 62

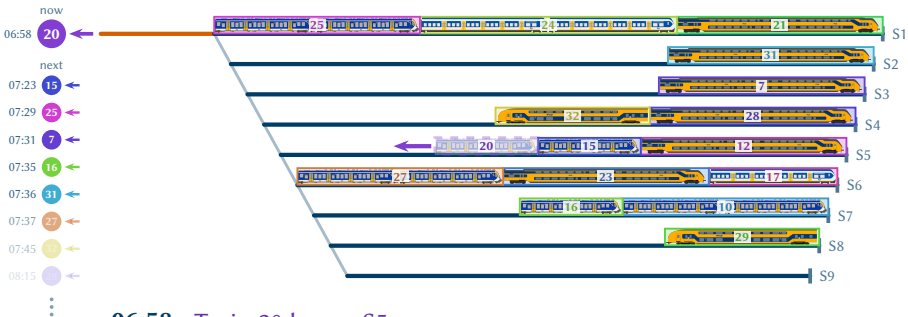
The plan, event by event



06:48 Train 26 leaves S5.

16 trains, 2220 m parked event 46 of 62

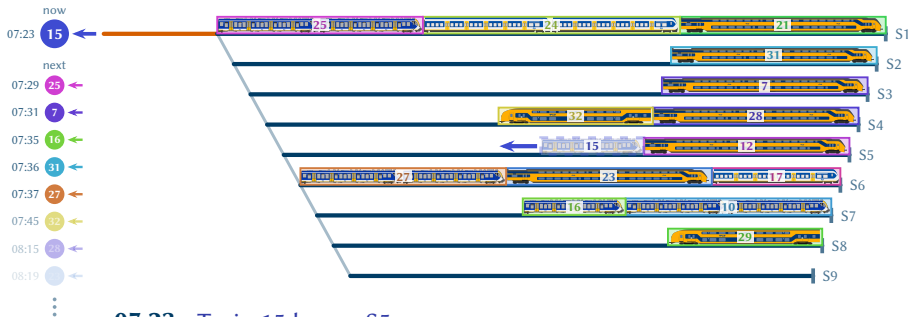
The plan, event by event



06:58 Train 20 leaves S5.

15 trains, 2140 m parked event 47 of 62

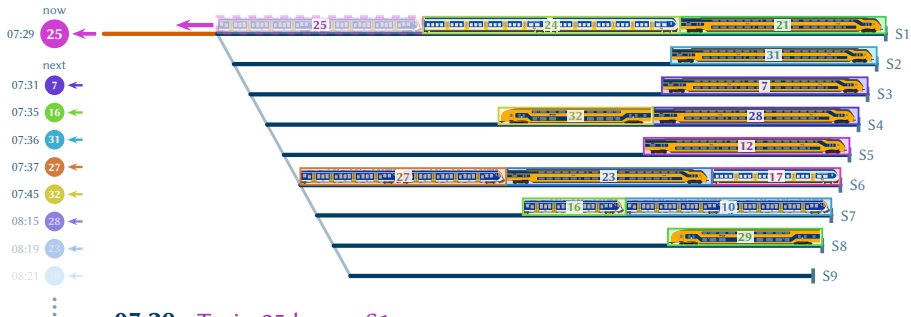
The plan, event by event



07:23 Train 15 leaves S5.

14 trains, 2060 m parked event 48 of 62

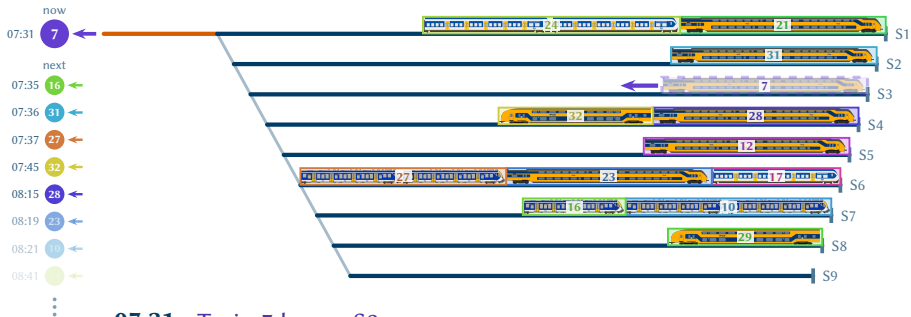
The plan, event by event



07:29 Train 25 leaves S1.

13 trains, 1900 m parked event 49 of 62

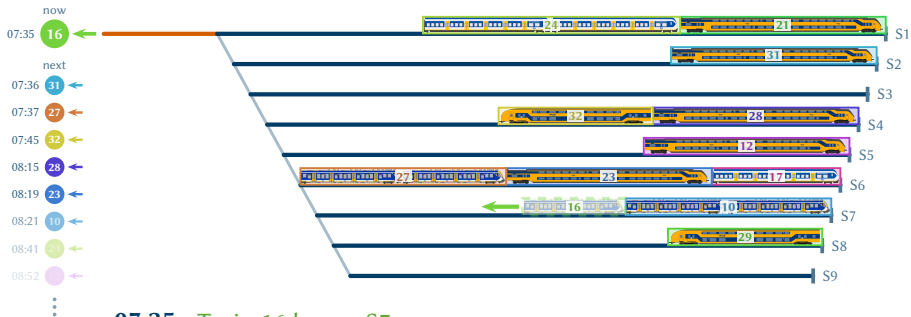
The plan, event by event



07:31 Train 7 leaves S3.

12 trains, 1740 m parked event 50 of 62

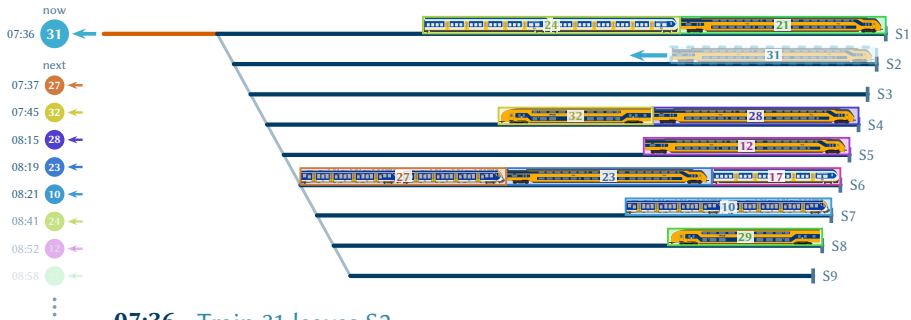
The plan, event by event



07:35 Train 16 leaves S7.

11 trains, 1660 m parked event 51 of 62

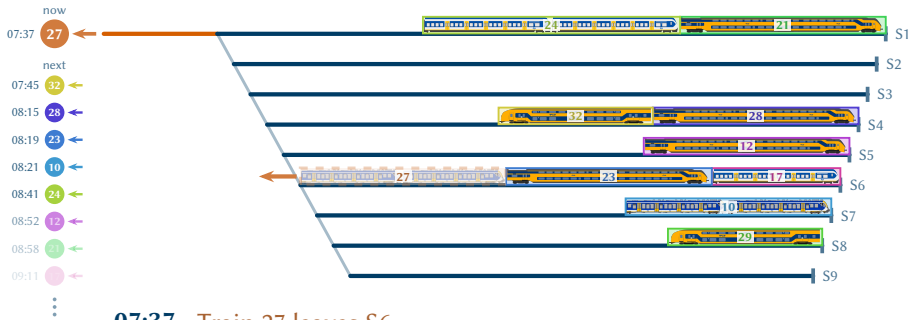
The plan, event by event



07:36 Train 31 leaves S2.

10 trains, 1500 m parked event 52 of 62

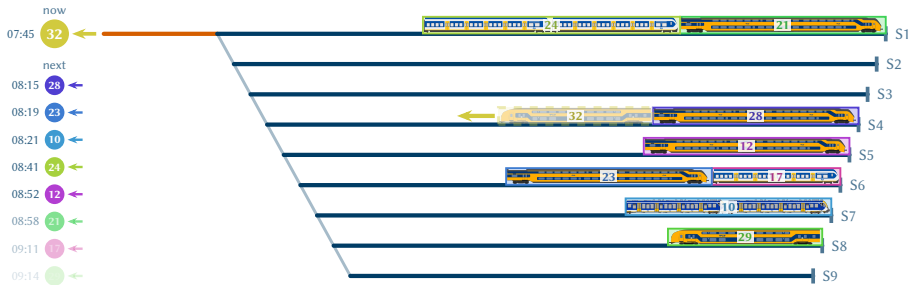
The plan, event by event



07:37 Train 27 leaves S6.

9 trains, 1340 m parked event 53 of 62

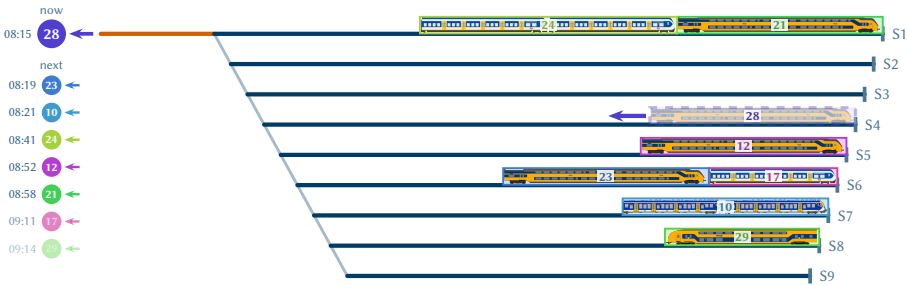
The plan, event by event



07:45 Train 32 leaves S4.

8 trains, 1220 m parked event 54 of 62

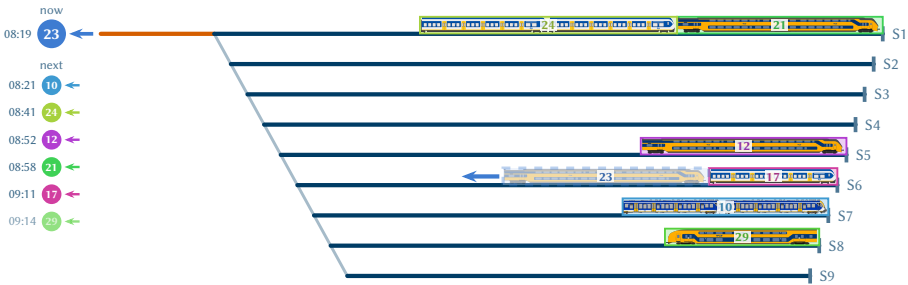
The plan, event by event



08:15 Train 28 leaves S4.

7 trains, 1060 m parked event 55 of 62

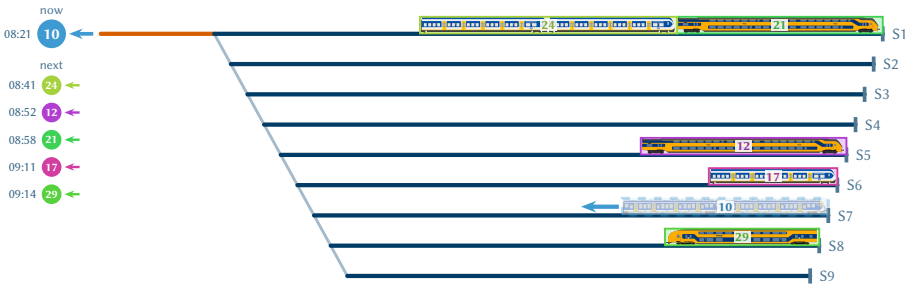
The plan, event by event



08:19 Train 23 leaves S6.

6 trains, 900 m parked event 56 of 62

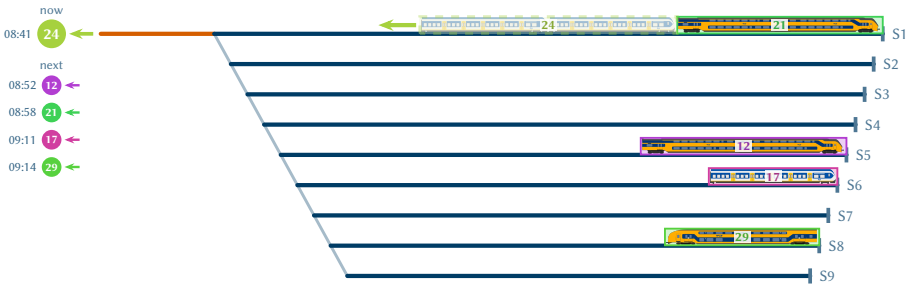
The plan, event by event



08:21 Train 10 leaves S7.

5 trains, 740 m parked event 57 of 62

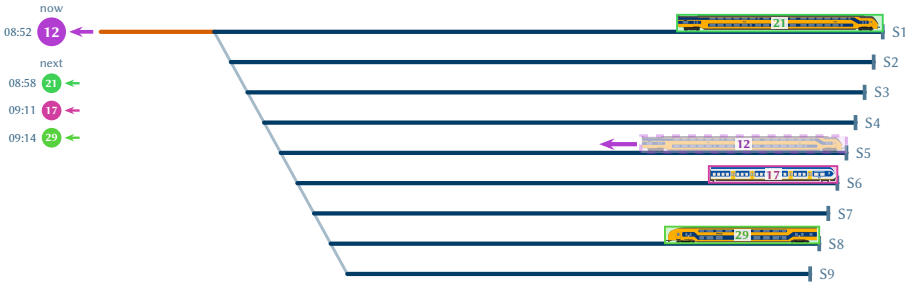
The plan, event by event



08:41 Train 24 leaves S1.

4 trains, 540 m parked event 58 of 62

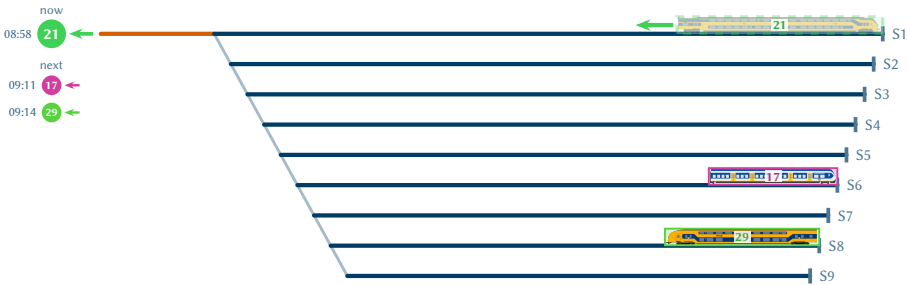
The plan, event by event



08:52 Train 12 leaves S5.

3 trains, 380 m parked event 59 of 62

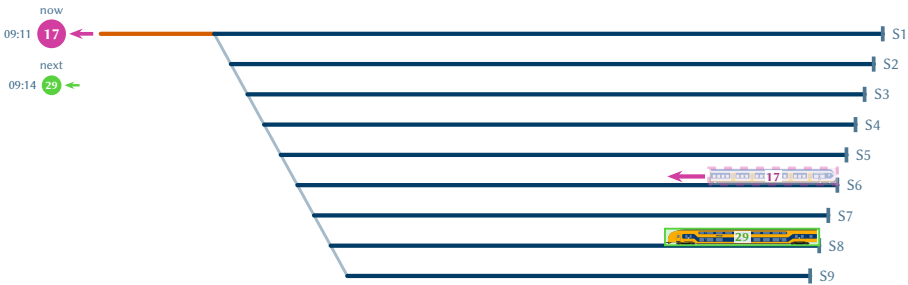
The plan, event by event



08:58 Train 21 leaves S1.

2 trains, 220 m parked event 60 of 62

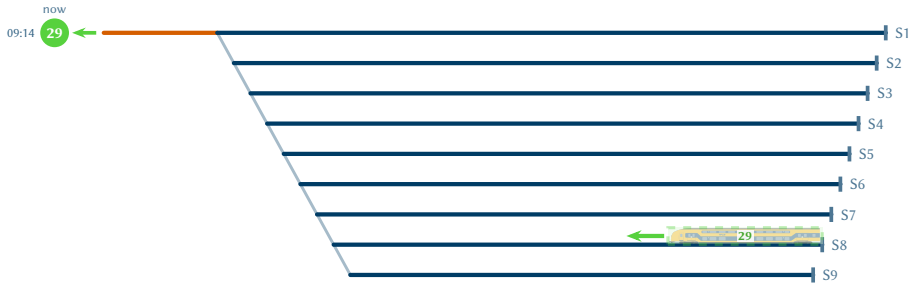
The plan, event by event



09:11 Train 17 leaves S6.

1 train, 120 m parked event 61 of 62

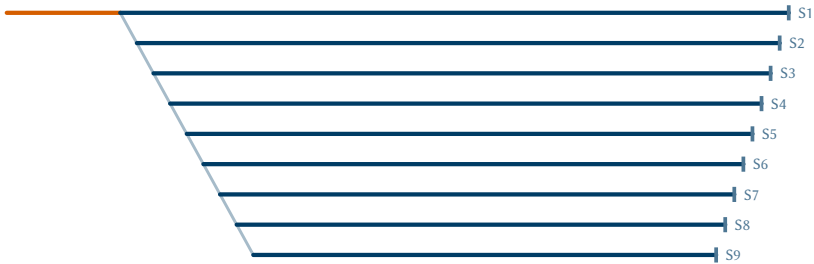
The plan, event by event



09:14 Train 29 leaves S8.

0 trains, 0 m parked event 62 of 62

The plan, event by event



09:14 The last train has left. The depot is empty again.

PART 5

Real depots

Partners and collaborators

Stabling research at SINTEF: the OYO project, 2025–2027.

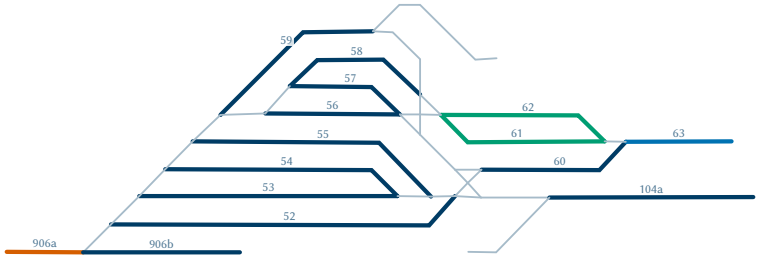
- An innovation project of **Siemens Mobility**, funded by the **Research Council of Norway**.
- Goal: new algorithms for efficient stabling and shunting in train yards.
- Tested on data from real yards: Kleine Binckhorst (NS, The Hague), Dortmund (DB) and Ryen (Sporveien, Oslo).

SIEMENS

The Research Council of Norway

Kleine Binckhorst, The Hague

Yard data published by NS and TU Delft (Robust-Rail).

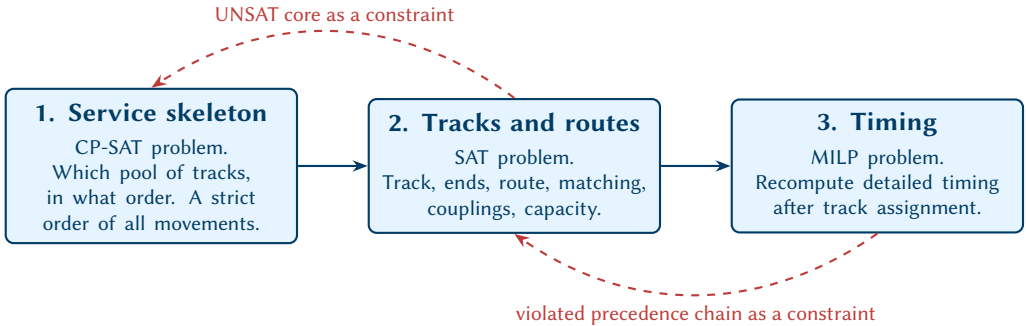


- Some tracks are open at both ends.
- Track 104a is reached only *through* one of 52 to 55, and 60 to 63 only through 56 to 59.
- Departures ask for unit types, not for specific trains.
- Trains are cleaned, washed, split and coupled, and plans are ranked by an objective.

What the formula leaves out

Two-ended tracks	Variables for which end each train uses. Then a train may fit only if parked trains move, so we must also check that every train can keep one spot.
Routing	Route variables. A route is blocked by any train on a track it passes through.
Matching	Which unit serves which departure. Departure times become variables.
Splitting, coupling	Trains can be split and recombined, for example to clean the units separately. When and how to do it is a decision in the plan.
Cleaning, washing, maintenance	A platform takes one train at a time, so the services must be put in order.
Drivers, cleaning crews	Every movement needs a driver and every cleaning a crew, and only a few are on duty at a time.

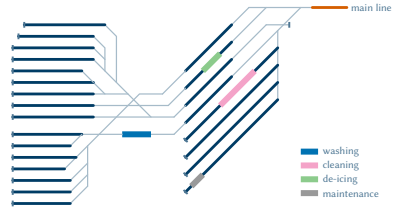
A more comprehensive solver: three stages



- Stage 1 fixes the order of all movements, so stage 2 needs no variables for times.
- Stage 2 is today's formula with the extensions of the last slide. Stage 3 computes the detailed timing with a MILP, and sends a chain of precedences that does not fit back to stage 2.

Takeaways

1. **Mathematical modeling.** Describe the problem precisely, with variables and constraints. Then a general solver, here **SAT**, does the search.
2. **The encoding decides what is fast.** The same problem can be written in many ways and understanding the solver is important.
3. **Problems with no solution.** The solver can prove that no plan exists, and find out why.
4. **Real-world problems are complex.** Routes, services, crews. To handle them we can extend the *model* instead of inventing a new algorithm, which is often easier and gives more insight.

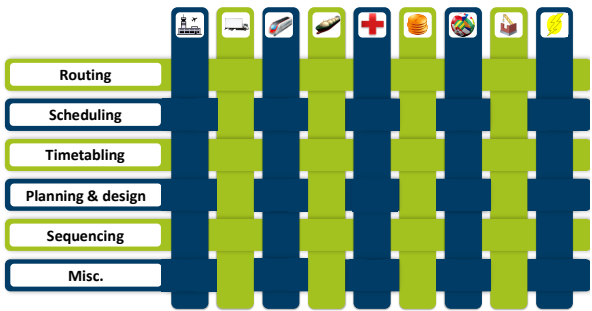


Which train goes on which track? Write the formula, let the solver search, and ask it why when the answer is no.

SINTEF Optimization

SINTEF Digital, Mathematics and Cybernetics, Oslo.

We do research and work with industry partners on **hard optimization problems** in routing, scheduling, timetabling, planning and design, for railways, logistics, maritime transport, health and energy.



Master's topics and summer internships are available.

bjornar.luteberget@sintef.no

SINTEF



Technology for a better society