Planning shunting operations at railway hubs

Han Hoogeveen

Joint work with Marjan van den Akker, Roel van den Broek, Rens de Heer, Ad Feelders, Jippe Hoogeveen

Scheduling seminar November 26, 2025

Contents of this talk

- Planning the operations at a single shunting yard (service site).
- Planning the operations in the entire station area.
- Scheduling the train drivers.

If you have a question during the talk, please speak up!

Slides are not extremely fancy: we don't care about the color of the hood, but about what is running underneath.

How it all began

- I met Bob Huisman (NS = Dutch Railways) at the MISTA conference in Prague, 2015.
- Bob told me about the planning problems they faced at the shunting yards.
- Extremely difficult: ROADEF challenge, Lex Schrijver worked on it (result not good enough)

How it all began

- I met Bob Huisman (NS = Dutch Railways) at the MISTA conference in Prague, 2015.
- Bob told me about the planning problems they faced at the shunting yards.
- Extremely difficult: ROADEF challenge, Lex Schrijver worked on it (result not good enough)
- My suggestion: this problem can be solved using Local Search
- Okay, let's do that.

Next steps

- I found a clever master student to do the job for me: Roel van den Broek.
- Roel graduated cum laude; got the Vliegenthart prize for his work.

Roel van den Broek

Next next steps

- Roel got a PhD-position financed by NS.
- Roel got his PhD and works now for NS.
- On basis of Roel's algorithm, NS has developed the program HIP (Hybrid Integrated Planning) to solve these shunting problems; HIP has gone life recently
- It led to structural cooperation with NS within the Kickstart AI initiative: I am working one day a week now at NS (secondment).
- And it played an important role in the start of the AI & Mobility Lab at UU.

AI & Mobility Lab UU

- We develop Innovative AI techniques to solve Societal Challenges in the domain of Public Transport and Logistics.
- Not only in the fields of Operations Research, Data and Machine Learning, but also Human Computer Interaction; we cooperate with Social Sciences.
- Our partners are companies like NS, ProRail, AirFrance-KLM, and Qbuzz.
- 15 PhD students including ROBUST Rail Lab (with TU Delft, NS, ProRail)
- This all started with a chat and a drink at MISTA, 2015.

Time for a commercial break!

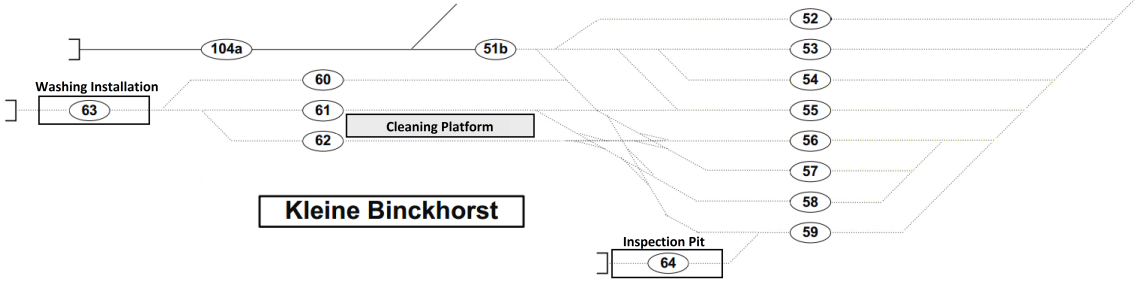
MISTA 2027 Tours

- MISTA will be back!
- June 7-11, 2027: MISTA will be held in Tours (France).
- Main organizer: Vincent 't Kindt
- Website: https://www.mistaconference.net/

Planning at service locations for Netherlands Railways

- Not needed material is stored at service sites (shunting yards)
- Here the train units are inspected; small repairs possible
- Some train units are cleaned internally and/or washed
- In the morning the train units leave according to the time-table in the pre-specified composition

An example of a service site



What do we have to deliver?

A **shunting plan** in which

- each departing train leaves the service site on time and in the correct composition;
- all service tasks are completed by skilled staff, at an allowed location, and without exceeding the limits of the resources;
- no parking, routing or resource conflicts occur.

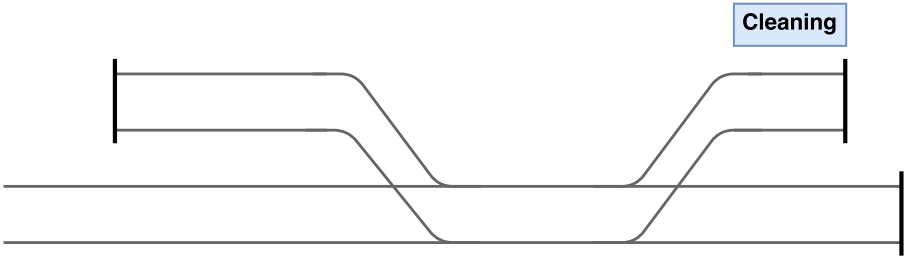
What makes the problem hard

- Interlinked parts:
  - Matching arriving and departing train units
  - Combining and splitting
  - Parking
  - Routing
  - Service scheduling
- Highly occupied location (up to 90%)
- Intermediate moves are needed

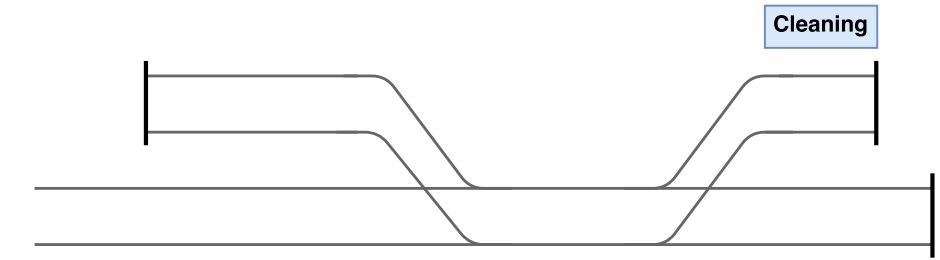
Challenge

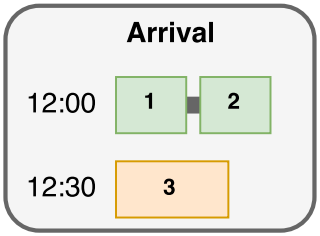
Find a feasible, efficient, and robust shunting plan.

Example

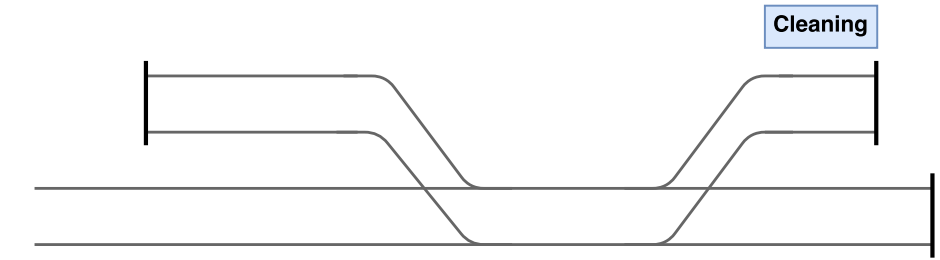


Example

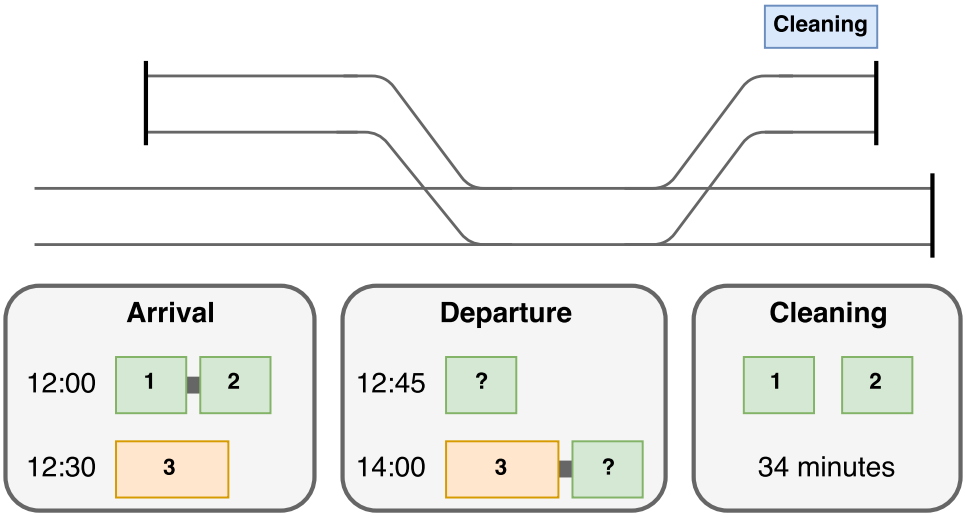




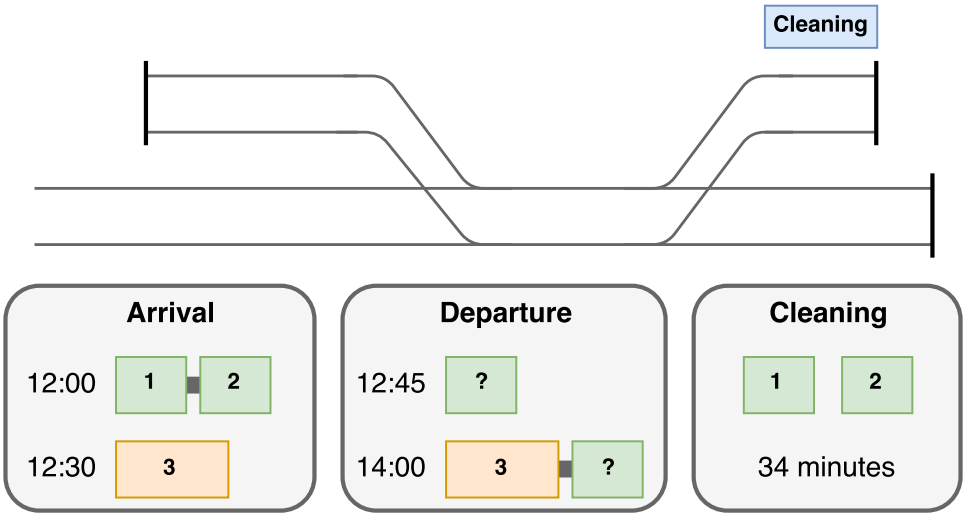
Example



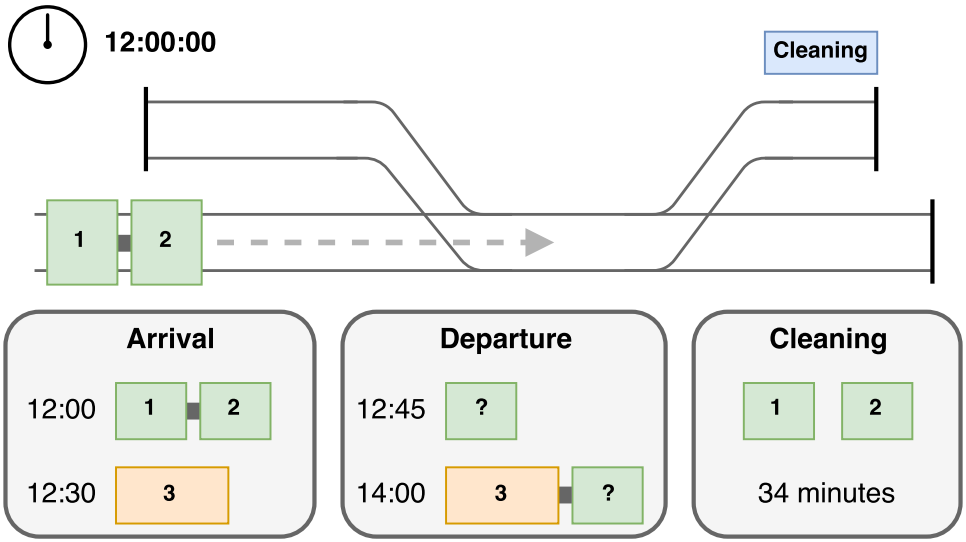
Example


Cleaning
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

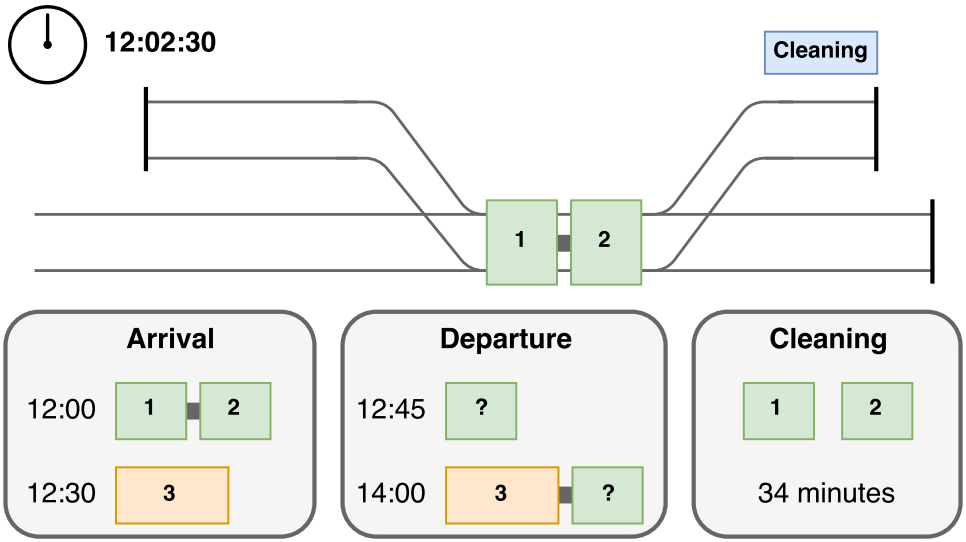
Example


Cleaning
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

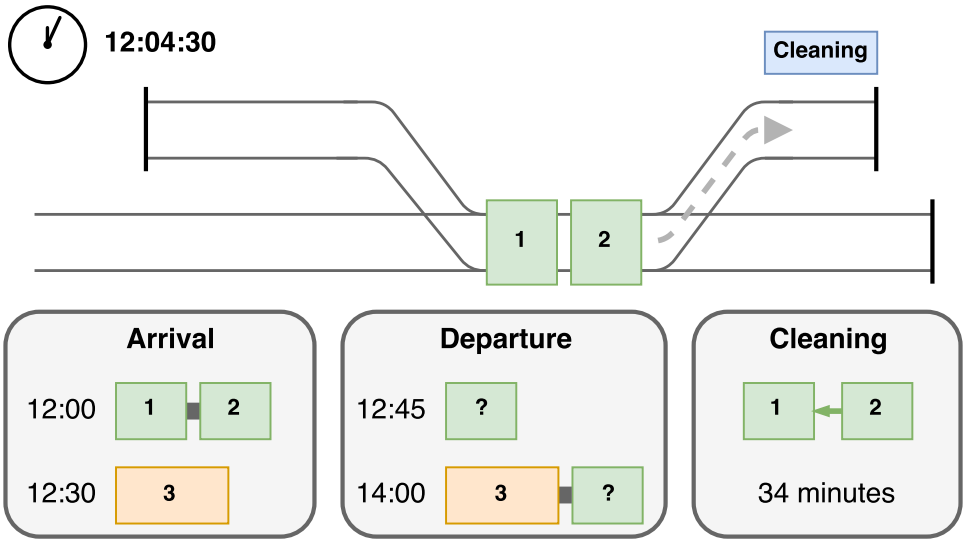
Example


12:00:00
Cleaning
1
2
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

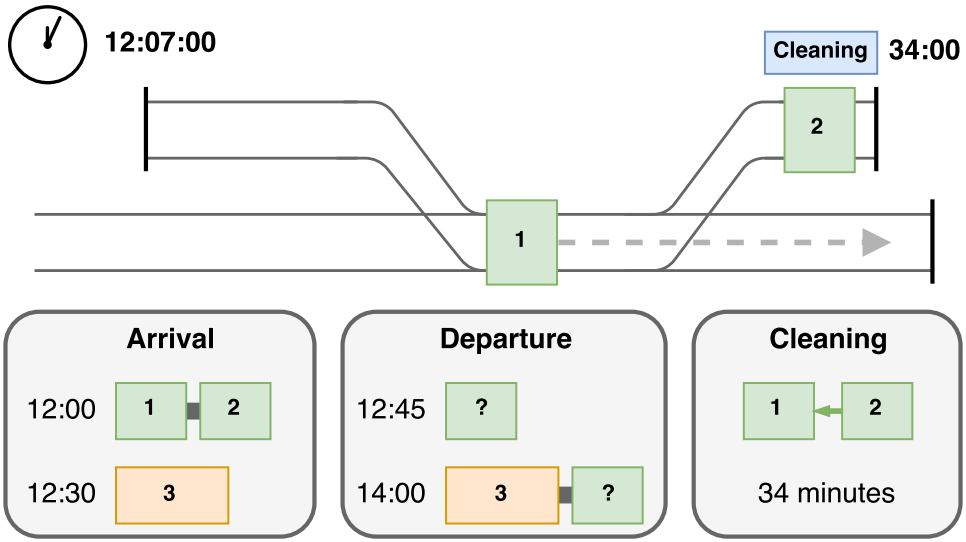
Example


12:02:30
Cleaning
1
2
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

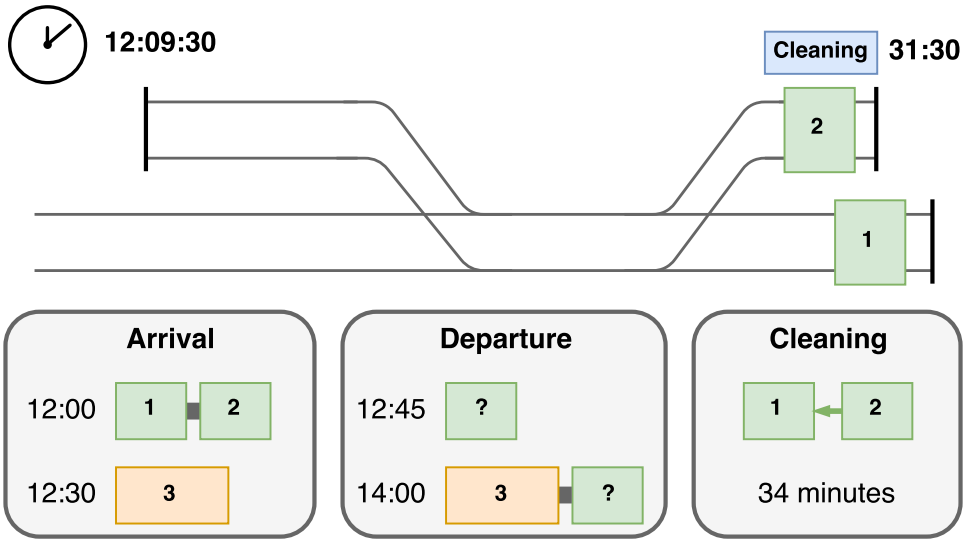
Example


12:04:30
Cleaning
1
2
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

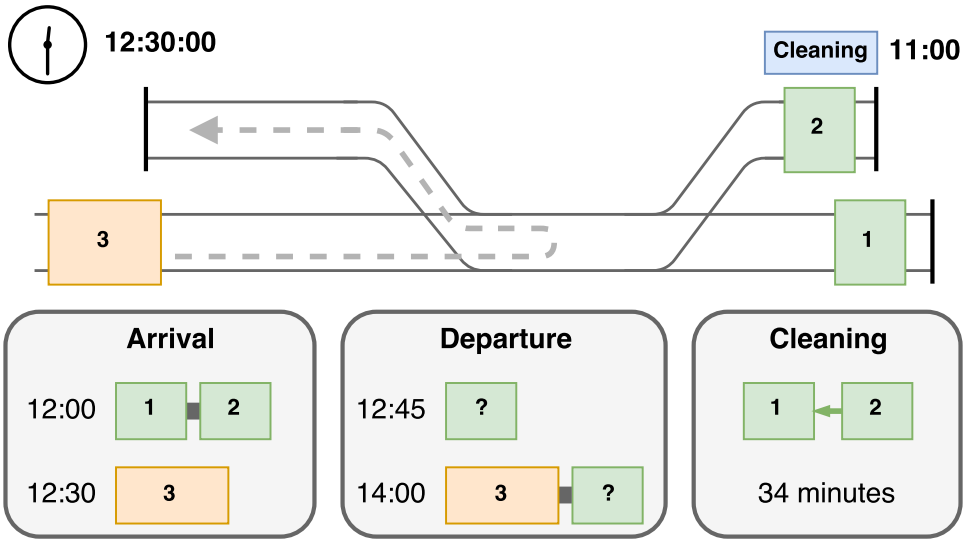
Example


12:07:00
Cleaning
34:00
2
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

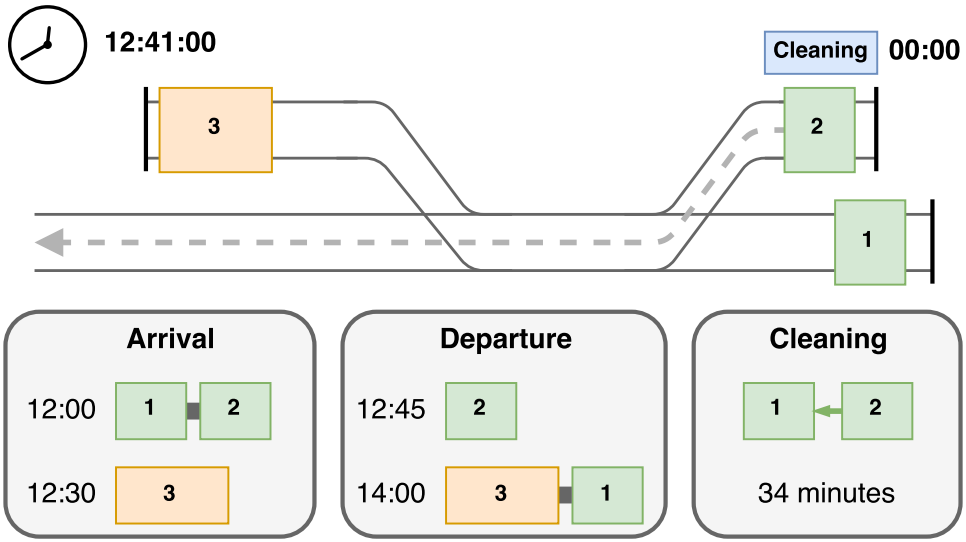
Example


12:09:30
Cleaning
31:30
2
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

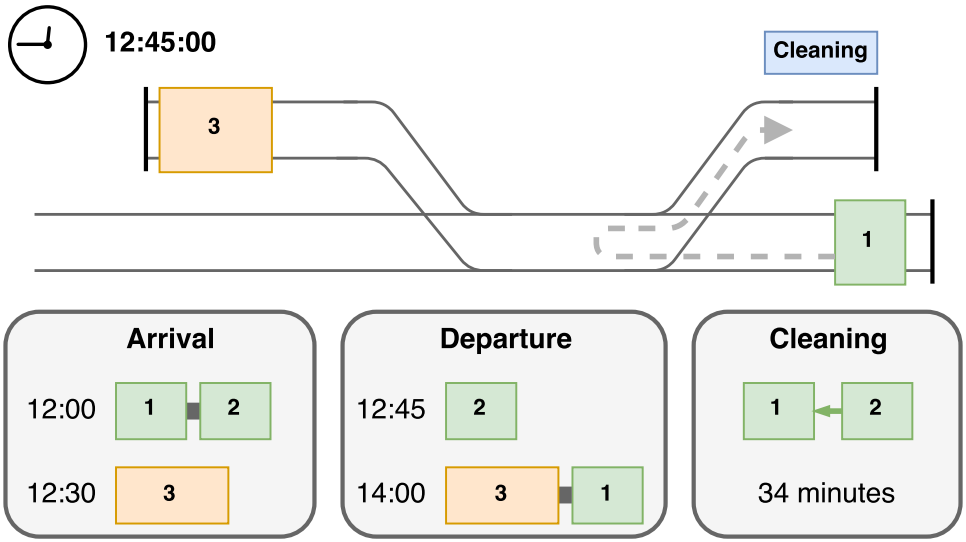
Example


12:30:00
Cleaning
11:00
2
3
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
?
14:00
3
?
Cleaning
1
2
34 minutes

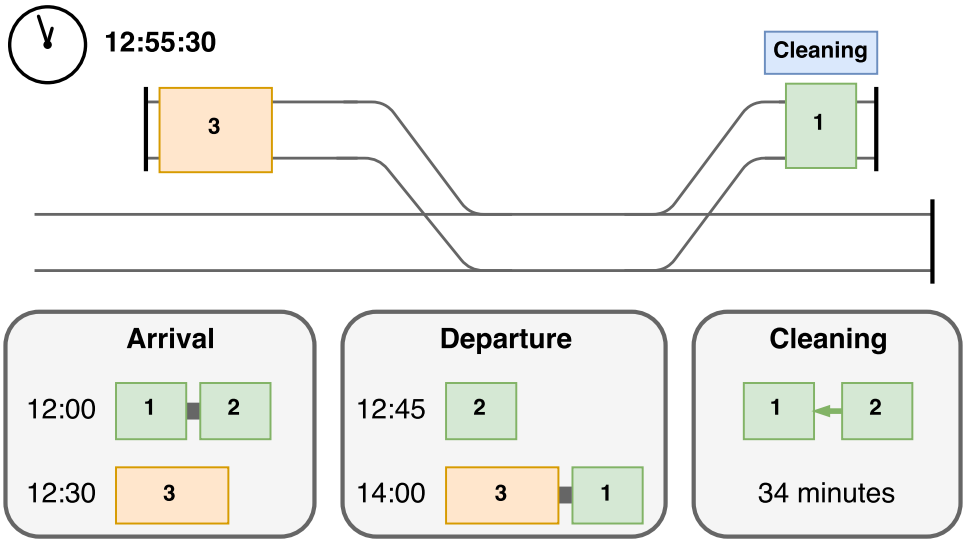
Example


12:41:00
Cleaning
00:00
3
2
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

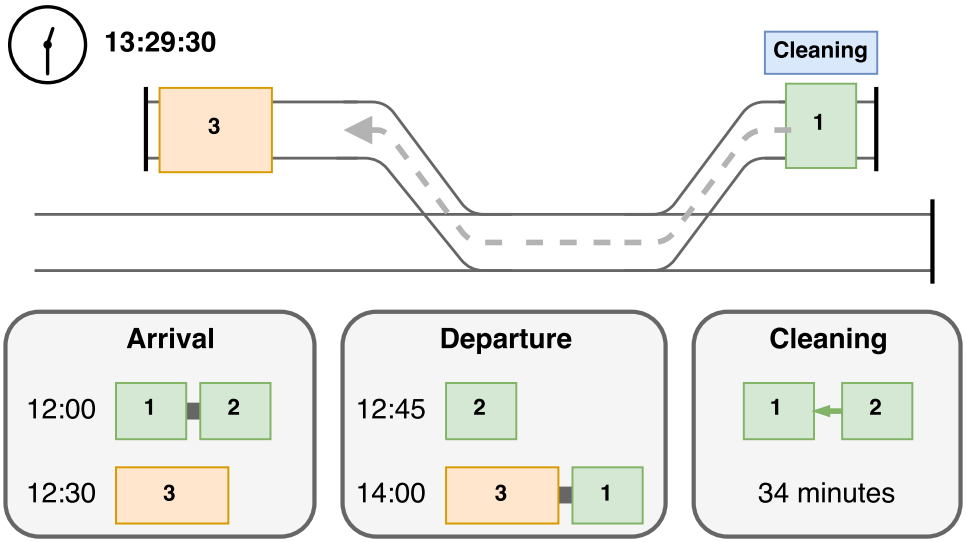
Example


12:45:00
Cleaning
3
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

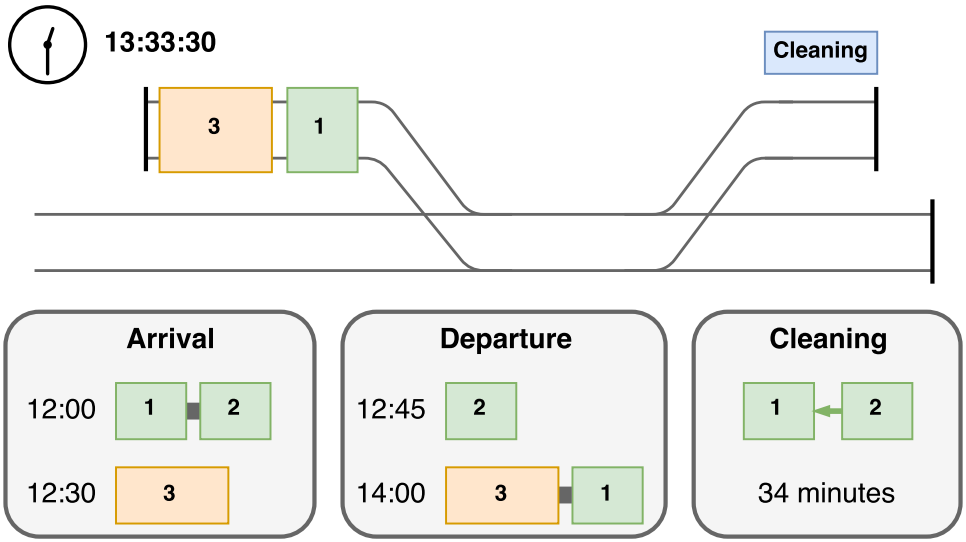
Example


12:55:30
Cleaning
3
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

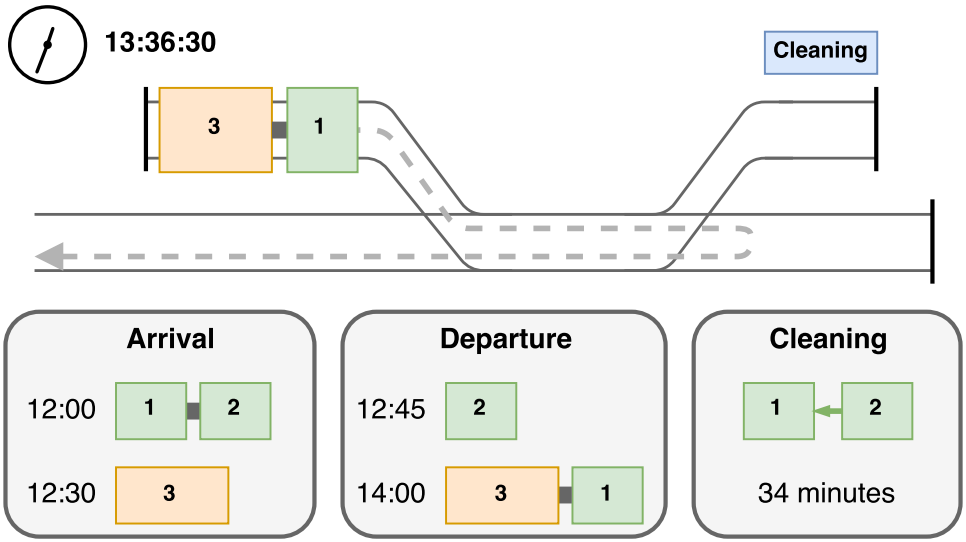
Example


13:29:30
Cleaning
3
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

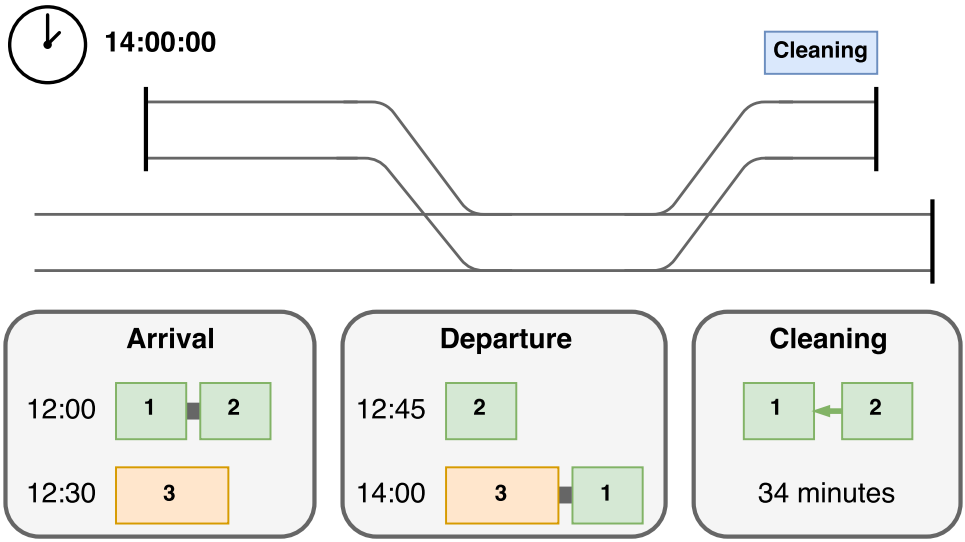
Example


13:33:30
Cleaning
3
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

Example


13:36:30
Cleaning
3
1
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

Example


14:00:00
Cleaning
Arrival
12:00
1
2
12:30
3
Departure
12:45
2
14:00
3
1
Cleaning
1
2
34 minutes

An integrated local search approach

We have to solve the following problems:

- Matching arriving and departing train units
- Combining and splitting
- Parking
- Routing
- Service scheduling

The personnel is planned later; this works fine.

An integrated local search approach

We have to solve the following problems:

- Matching arriving and departing train units
- Combining and splitting
- Parking
- Routing
- Service scheduling

The personnel is planned later; this works fine.

That was some years ago; now it is more of a problem (final part of the talk).

Constraints and Objective Function

Does there exist a feasible plan?

Transform the decision problem to an optimization problem by relaxing some of the feasibility constraints.

Relaxed Constraints

- Delayed departures
- Track overflows
- Blocked movements

Hard Constraints

- Solutions are always resource feasible
- The train matching is valid

The severity of conflicts and the number of movements are included in the objective to guide the local search.

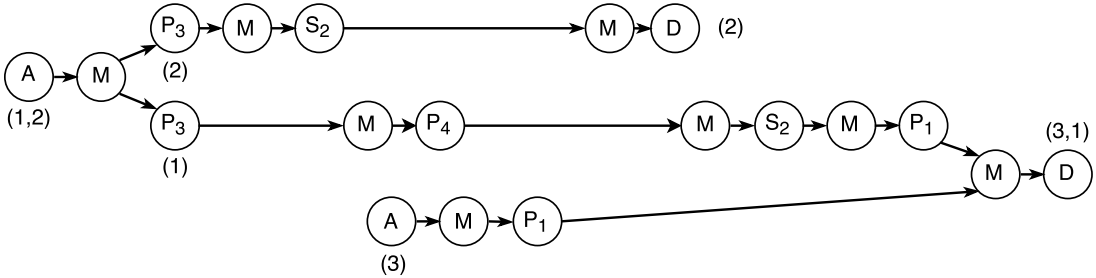
Plan Representation

An activity plan is represented as a *Partial Order Schedule*. The model consists of a precedence graph with five types of nodes:

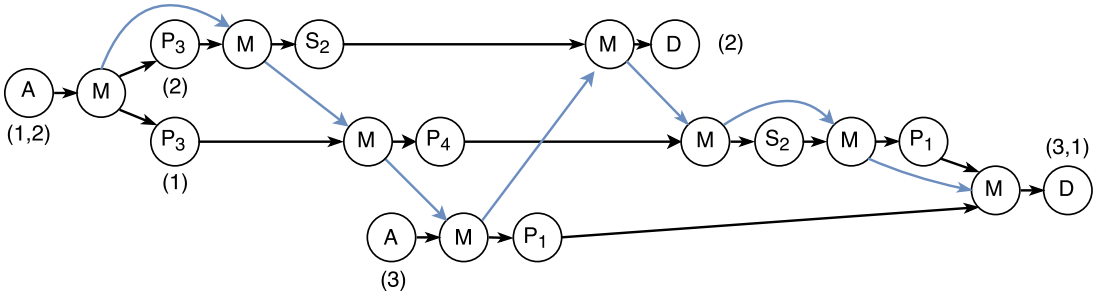
1. Arrival (A)
2. Departure (D)
3. Movement (M)
4. Service (S)
5. Park (P)

The arcs represent precedence relations between the different nodes.

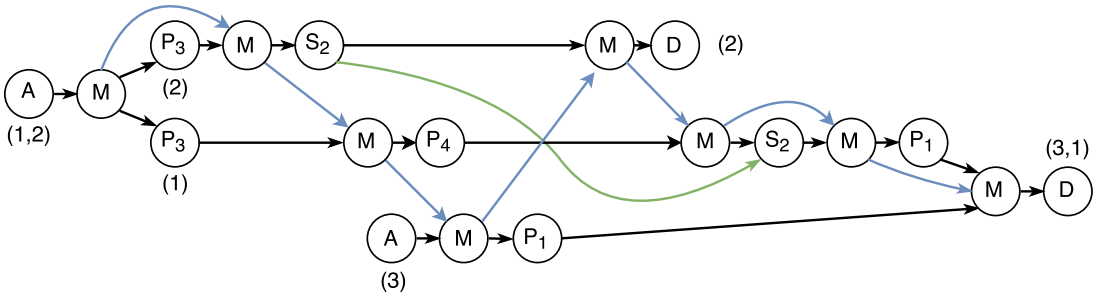
Activity Graph (corresponding to the example)



Activity Graph (corresponding to the example)



Activity Graph (corresponding to the example)



The model is evaluated at each iteration to compute the movement paths and determine the conflicts.

Local Search Neighborhoods

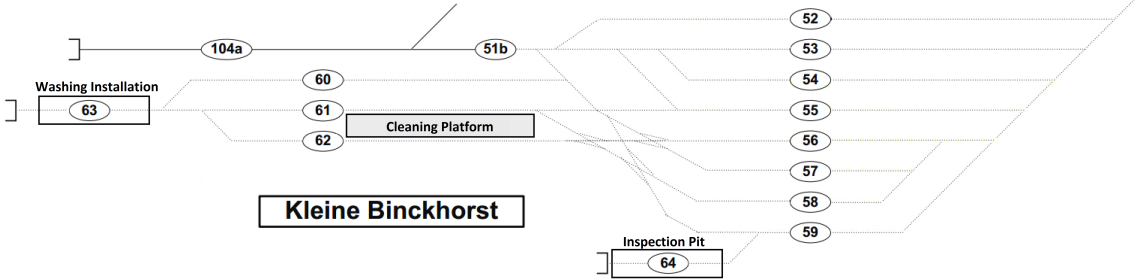
We alter a solution by

- shifting, adding or removing movements;
- changing the location of a parking activity to a different track;
- swapping the order of consecutive service tasks of the same resource / train;
- assigning a service task to a different resource;
- swapping the assignment of (partial) trains in the current matching.

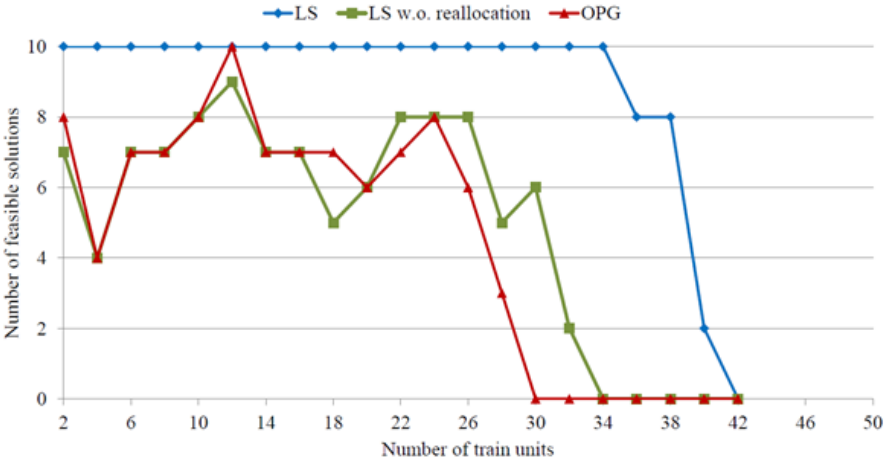
Experiments

- Tested on The 'Kleine Binckhorst' (service site near The Hague)
- Generated 10 instances for each fixed number of arriving and departing train units.
- Arrival time, departure time and train composition drawn from empirical distribution based on real-world data.

Kleine Binckhorst



Kleine Binckhorst



Up to the second part

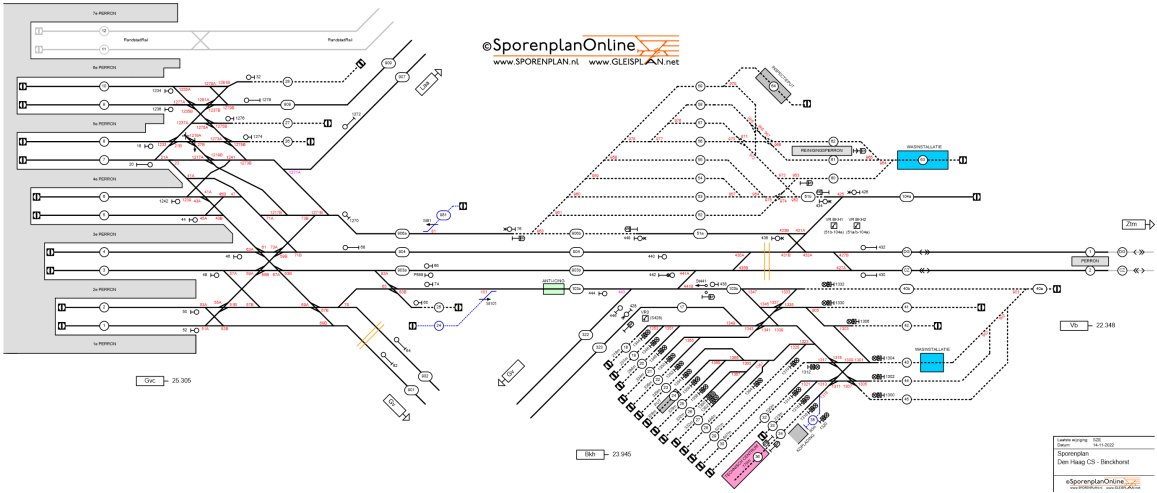
Any questions with respect to solving the single shunting yard problem?

Up to the second part

Any questions with respect to solving the single shunting yard problem?

Second part: Planning the entire station area.

Station area The Hague



Planning the entire station area

Main problem

You must distribute the rolling stock over the yards, such that

1. Good routes to and from the yards exist.
2. The yards can handle the train units that are sent there.

Preliminaries

- We can find routes using Constraint Programming. Some constraints:
  1. Some platforms are connected to only one yard (avoiding detours).
  2. There are (many) regular trains crossing the area; robustness is appreciated.
- We use HIP to judge whether an instance of the shunting problem is feasible; this may take some time.

Sequential approach? Not possible

- Route first and check feasibility next: this may require a large number of HIP checks.
- Distribute first and route next: if routing is possible, then the routes can be bad.

Conclusion

We need to apply an integrated approach.

Integrated approach

- Checking feasibility with HIP each time is not possible.
- We need a quick computational rule to estimate the probability that the instance is feasible.
- This check must be incorporated within the CP that finds the routes to steer towards feasible distributions.

We need a (preferrably linear) classifier to estimate the feasibility of a distribution of the rolling stock over the shunting yards

Classifier

Important aspects for deciding feasibility:

1. Length and number of train units
2. Service requirements
3. Number of splits and combines

Use Explainable AI to find this classifier

- Assumption: instance feasible if and only if HIP solves it
- We want to predict whether HIP can solve it to feasibility
- Apply Logistic Regression (form of Supervised Learning)
- Training on basis of 2000 test instances

Logistic regression

- Estimate a linear function $L(x)$, where x is the input
- This function differs per shunting yard
- Probability of success:

$$P(x) = \frac{1}{1 + e^{-L(x)}}$$

Formula for the classifier

Constant (22.975) minus total contribution per

- Train Type (0.844 per SLT4 etc.)
- Combined Arrivals (-0.197 per SLT etc.)
- Combined Departures (-0.205 per SLT etc.)
- Service request (0.132 per SLT etc.
- Splits (0.138)
- Combines (0.296)

Combining this with the routing CP (1)

- We use Constraint Programming (CP) to find routes
- In this CP we take into account the regular traffic in the station area
- Any solution to the CP yields a value for $L1(x)$ and $L2(x)$ (for shunting yards 1 and 2).

Combining this with the routing CP (2)

- We include in our objective function that we want to maximize the minimum of $L1(x)$ and $L2(x)$ to maximize the probability that HIP can solve both yards
- This approach finds good routes resulting in a good distribution of the rolling stock

Outcome

- We use the resulting distribution of the train units
- This allows us to decompose the problem and solve the problem for each yard separately.

Up to the third part

Any questions with respect to solving the station area problem?

Up to the third part

Any questions with respect to solving the station area problem?

Third part: Planning the machine drivers.

Planning the drivers in the station area

- NS uses ILP to plan the drivers
- The ILP cannot solve all instances => new approach needed
- Carried out by Jippe Hoogeveen as an experimentation project.
- Project in cooperation with NS department SENSOR
- The goal is to plan the machine drivers at the station area, who carry out the plans from HIP.

Plan delivered by HIP

- HIP determines the tasks that must be done
- For each task you know
  - the time it takes (processing time)
  - the time interval during which it must be done plus a preferred start time
  - the start place and end place
  - predecende constraints
- For some tasks you need two drivers => synchronization

Machine drivers

- Drivers are identical, but work in different shifts
- Drivers are entitled to a break (in the canteen).
- We consider a break as a special task; one break per shift.
- Drivers must walk in between tasks $=>$ substantial walking distances.
- To cope with disturbances, slack between two subsequent tasks is desired.

Machine drivers

- Drivers are identical, but work in different shifts
- Drivers are entitled to a break (in the canteen).
- We consider a break as a special task; one break per shift.
- Drivers must walk in between tasks => substantial walking distances.
- To cope with disturbances, slack between two subsequent tasks is desired.

Temporary relaxation:

- Obey the start and end tims of shifts, but ignore their connections.
- One shift can get more than one break (or zero breaks)

In case of shifts 0-8 and 2-10, one driver can work from 2-8 with two breaks.

Objective function

- You get a cost (task dependent) for each task that is skipped
- You get a cost depending on the difference between the real starting time and the preferred starting time.
- You get a cost when the slack between consecutive tasks is smaller than desired.
- You get a cost on basis of the total walking time.
- You get a bonus for each pair of preferred consecutive tasks.

Important observation

- If we know for each task its start time, then the problem boils down to an assignment problem.

Important observation

- If we know for each task its start time, then the problem boils down to an assignment problem.
- Construct a graph G where each task corresponds to a vertex.
- Add dummy vertices for the start and end times of the shifts.
- Include an arc (i, j) if task j can be done immediately after i by a driver.

Important observation

- If we know for each task its start time, then the problem boils down to an assignment problem.
- Construct a graph G where each task corresponds to a vertex.
- Add dummy vertices for the start and end times of the shifts.
- Include an arc (i, j) if task j can be done immediately after i by a driver.
- Each driver schedule corresponds to a path in the network.

Important observation

- If we know for each task its start time, then the problem boils down to an assignment problem.
- Construct a graph G where each task corresponds to a vertex.
- Add dummy vertices for the start and end times of the shifts.
- Include an arc (i, j) if task j can be done immediately after i by a driver.
- Each driver schedule corresponds to a path in the network.
- Determine for each task its successor $=>$ minimum weighted matching.
- To make it possible to skip task j include arc (j, j)

Decomposition approach: first attempt

- Split up the problem in two phases:
  1. Determine the start times of the tasks; use Local Search to make adjustments
  2. In each iteration determine the corresponding solution by solving the weighted matching problem.
- Given the final set of schedules, fix the shifts and the breaks (done once at the end).

Nice, new approach, but there is a complication:
it is cumbersome to adjust the start times using Local Search

Beautiful idea by Jippe: split Phase 1.

Improved decomposition approach

1. Determine which tasks can be done consecutively (graph G); use Local Search to make adjustments
2. Determine the corresponding driver schedules by solving the weighted matching problem (use the start times in the current solution to compute the costs).
3. Compute the best set of start times for these schedules using Linear Programming (LP)

Improved decomposition approach

1. Determine which tasks can be done consecutively (graph G); use Local Search to make adjustments
2. Determine the corresponding driver schedules by solving the weighted matching problem (use the start times in the current solution to compute the costs).
3. Compute the best set of start times for these schedules using Linear Programming (LP)

Before the next Local Search step, repeat until convergence:

- Given the determined starting times from the LP, update graph G
- Solve the weighted matching problem
- Run the LP to find the best set of starting times, etc.

Initial heuristic solution

- Initially let G include arc (i, j) if the maximum time difference is big enough (minimum start time for i, maximum start time for j).
- Solve the resulting matching problem

Initial heuristic solution

- Initially let G include arc (i, j) if the maximum time difference is big enough (minimum start time for i, maximum start time for j).
- Solve the resulting matching problem
- Determine new starting times through LP ($+$ random starting times for skipped jobs)
- Adjust G: the available time between i and j is determined as a weighted combination of the LP starting time differences and the maximum time differences $=>$ arc (i, j) may get removed if there is not enough time in between tasks i and j in the LP
- During the process the weight of the maximum time differences goes to zero.

Remaining details

- We use six different Local Search neighborhoods
- We construct a correct solution for the problem with shifts and breaks using ILP
  - The starting times of the tasks are fixed at the values of the relaxed problem
  - Next, bad arcs are removed (on basis of solution of the assignment problem)
  - Solve an ILP with variables $x_{s,i,j}$: shift s uses arc (i, j).
- Usually, this only increases the total walking times and not the number of skipped tasks.

Enhancements

- Jippe has achieved tremendous speed-ups by a smart implementation of the matching algorithm and the LP.
- NS has provided one test instances; originally many jobs had already been assigned to drivers.
- We have tested our algorithms on this instance (Amsterdam fixed = Ams F) and on the instance without the fixed assignments (Amsterdam unfixed = Ams U)
- Jippe further has generated a set of benchmark instances (example instance Difficult = Diff)

Results: outcome values

Method	Ams UF			Ams F		
	Value	Runtime	# Skipped	Value	Runtime	# Skipped
Preferred time	72,350	1.5	12	5,952,486	0.8	42
PS (ILP NS)	—	—	—	5,622,260	51	35
Full algorithm	-428,589	1.9	3	5,092,335	2.5	29
Optimal (ILP)	-429,800	23,832	3	5,089,236	18	29

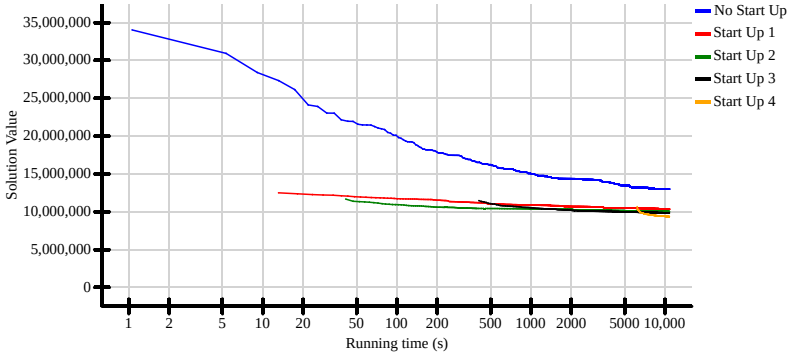
Table: Comparison on Ams UF and Ams F

Results: realized speed-up

Method	Ams F	Ams U	Diff
Fully optimized	1.2	1.4	80
Assignment optimized	8.1	10.8	223
Not optimized	27	79	2953

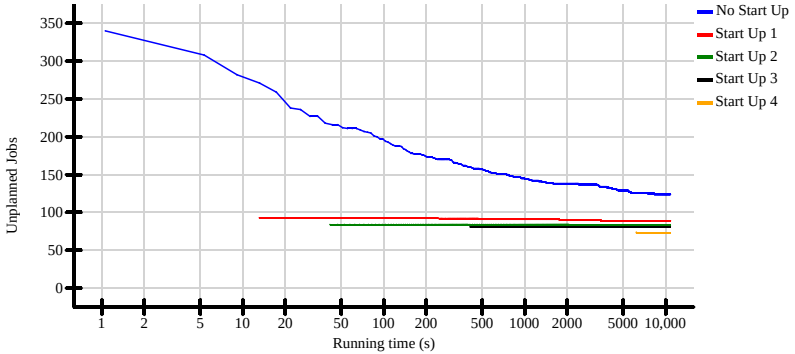
Table: Overview of the running times (seconds)

Effect of the initial heuristic for Diff instance: quality



Effect run time Start Up heuristic on solution value (colored line is run time Local Search)

Effect of the initial heuristic for Diff instance: unplanned jobs



Effect run time Start Up heuristic on number of skipped jobs

Future dreams

- HIP finds a POS for the events
- Using this, the times for the events are determined in a greedy way
- These times form the input of the driver planning => Sequential planning

It should be possible to get even better results by adjusting these time on basis of the output of Jippe's algorithm => Partly integrated planning on basis of the initial POS.

Outlook for solving the station area problem

1. Distribute the rolling stock over the yards: Logistic Regression and Constraint Programming
2. HIP solves the instances per shunting yard
3. Improve these HIP solutions by reducing the number of shunting movements (Algorithm by Rens de Heer)
4. Plan the drivers; if necessary adjust the start times.

Questions

A bright future lying ahead!