Rail yard shunting — A Challenge for CP?

Per Kreuger (piak@sics.se)
Joint work with Martin Aronsson

Swedish Institute of Computer Science (SICS)

11th March 2005

Outline

Outline

3 main types of transports

Postal services Fast point to point transports

System trains Services dedicated to large flows for particular customers

Car cargo pull in, shunting and distribution of cars at a number of shunting yards

- Rail yard shunting occurs mainly for third type
- Green Cargo is currently the only provider of the third type transport on rail in Sweden

Some figures

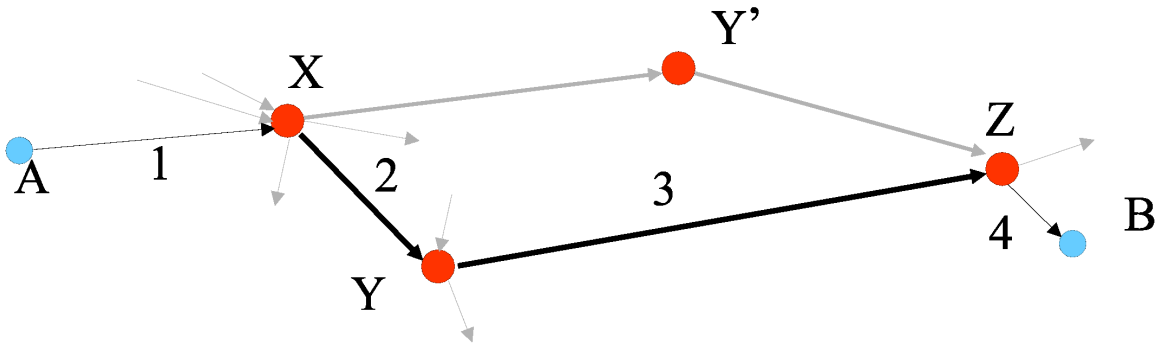
Green Cargo

- 420 engines for all types of transport
- 8500 cars + cars owned by customers
- 1-2000 "trains" per day (night)
- 13 000 million ton kilometres per year
- 6 billion SEK per year

For car cargo

- 25 shunting yards of varying sizes throughout Sweden
- Shunting yards typically handle 3-4 incoming trains hourly

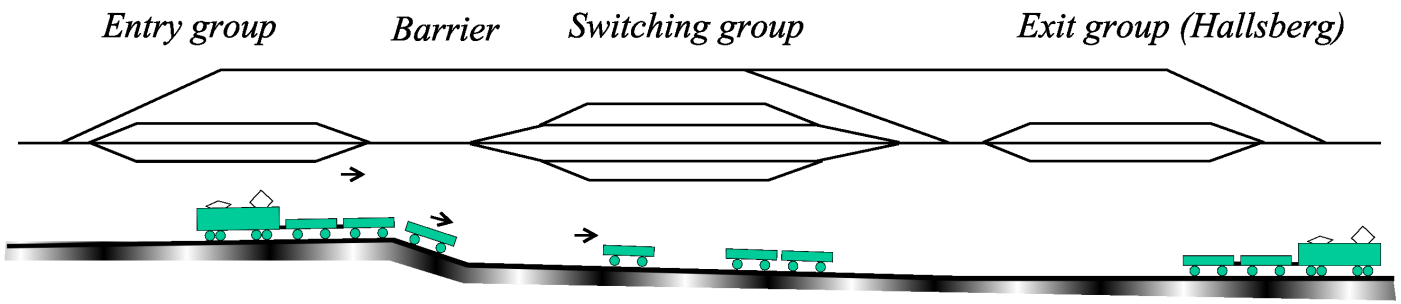
Car cargo system



- Individual cars are routed from end to end through rail network
- Cars are transported in trains that are assembled and disassembled at shunting yards
- Flow of cars through network varies from day to day depending on demand

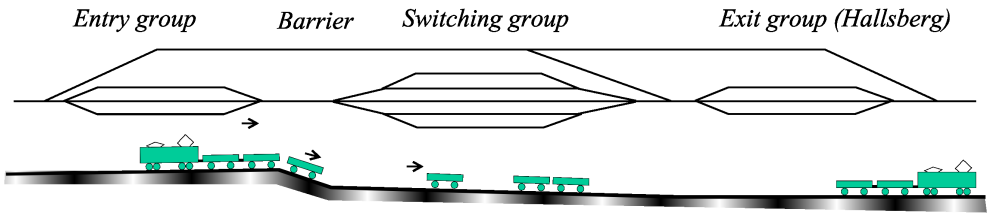
Outline

Rail yard shunting



- Incoming trains arrive at (more or less) fixed times but use the entry group as a buffer and preparation area
- Each incoming train is pushed over the *barrier* where cars are separated and roll down to the shunting group
- Switches are operated so as to distribute the cars into a number of destination tracks
- Under "*normal*" operation, a unique departing train is assembled at each destination track
- Assembled trains depart (if possible) at fixed times
  - The exit group (if present) is used for finishing

Capacity of shunting yard resources



- The entry group must accommodate incoming trains
- The *barrier* has a (more or less) fixed capacity
- The shunting group limits the number of trains simultaneously being assembled
- Each track in the shunting group is allocated to a departing train from the arrival of its first car until its scheduled departure; This can be *several hours*
- This is not feasible in practise: The capacity of the shunting group is in most cases too small at peak hours

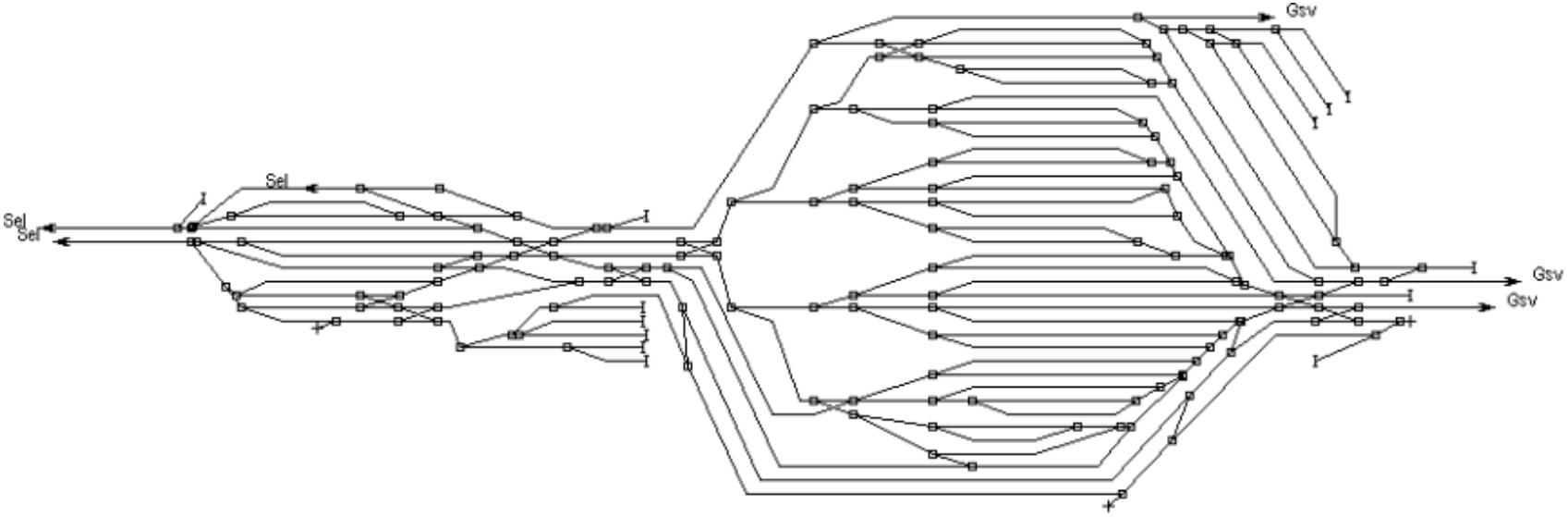
Solution: Build *temporary* "trains" that can be routed back to the entry group

Open problems

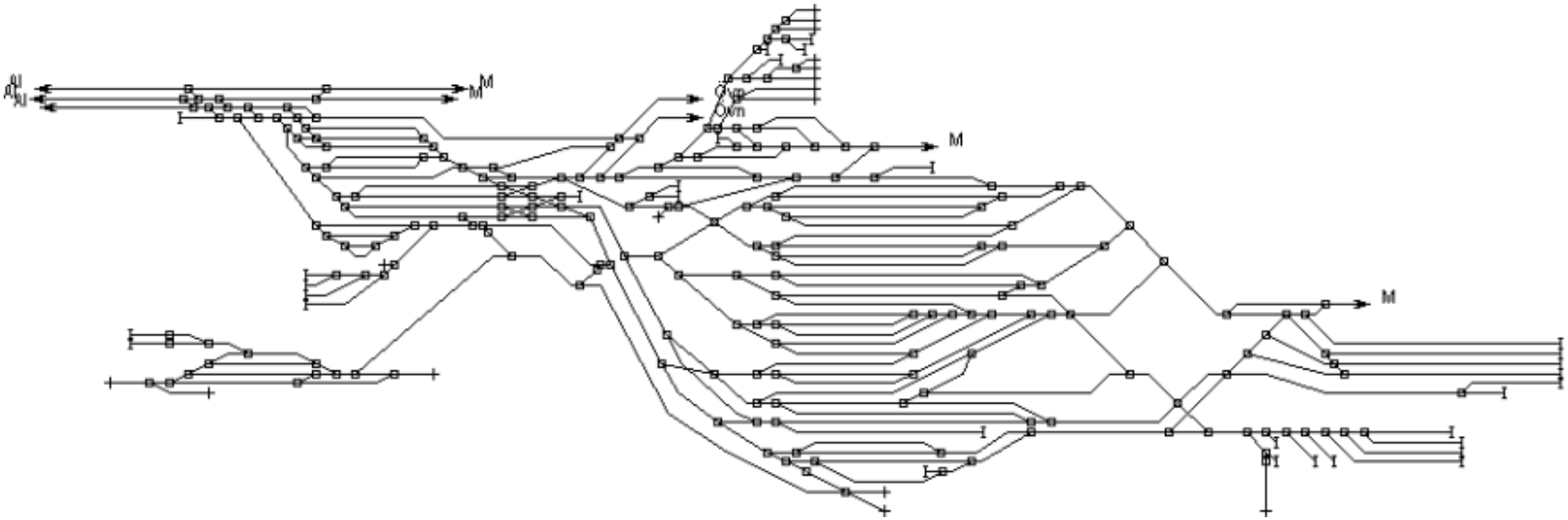
- Which cars should be combined into temporary trains?
- When should the temporary train be routed back to the entry group?
- When should the temporary train be pushed over the barrier?
- How is the problem best represented?
  - Flows?
  - Scheduling and capacity conditions?
- It is clear that we wish as few temporary trains as possible, but are there other cost measures?

Outline

Sävenäs shunting yard

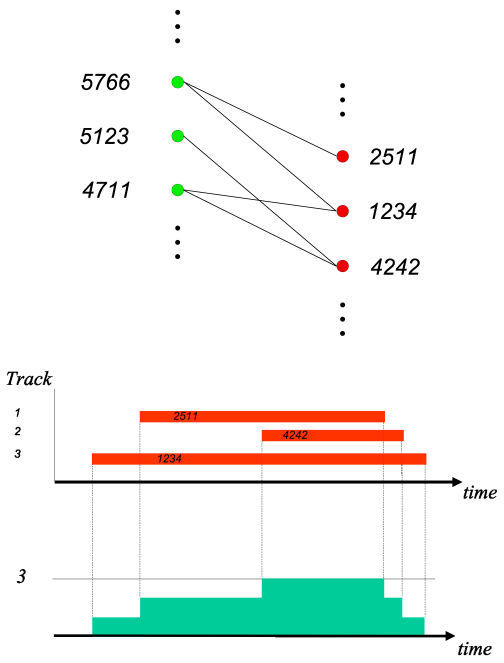


Malmö shunting yard



Outline

Without temporary trains



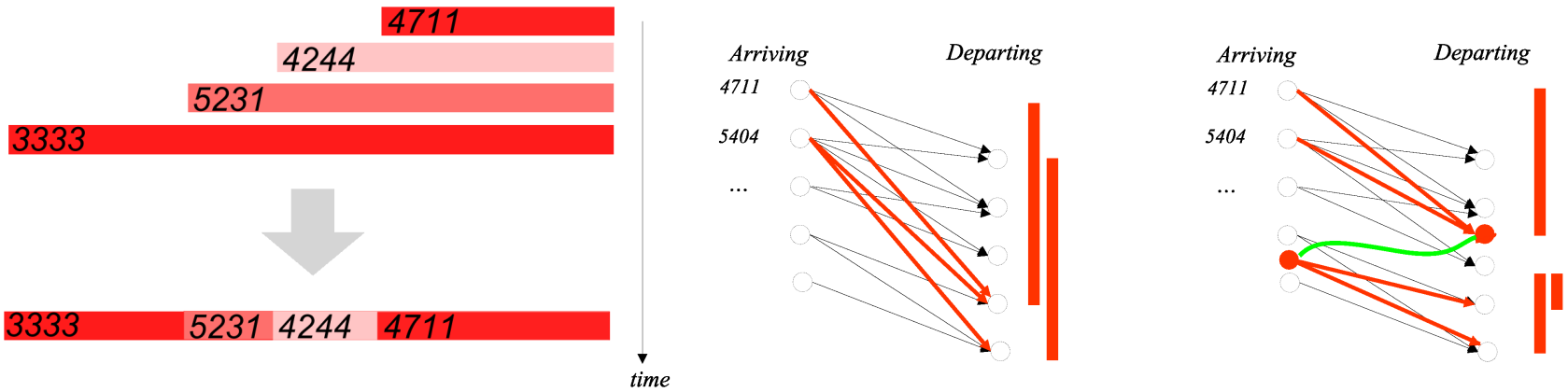
- The flow from incoming trains to outgoing trains is fixed
- Allocation of arriving trains to entry group tracks and of departing trains to shunting group tracks can be (more or less) disregarded
- Three resources need to be scheduled
  1. Entry group (cumulative)
  2. barrier (serialise)
  3. Shunting group (cumulative)

Durations:

- 40 min. at entry group
- 15-20 min. /train at barrier
- 60 min. at shunting group

Restriction: Depart. - Last arrival > 140 min.

With temporary trains



- Cars destined for distinct departing trains may be routed to a single track
- Capacity utilisation of shunting group is reduced
- Time to scheduled departure must be large enough to accommodate the additional duration required for the temporary train to be routed back to the entry group and over the barrier again
- The entry group and barrier must be able to accommodate additional tasks for the temporary trains

Outline

Flow matrix without temporary trains

	A	B	C	D	E	F	G
1	1	1	1	1			
2	1	1	1	1			
3		1	1		1	1	
4		1	1	1		1	
5				1	1		1
6					1	1	
7						1	1

- All flows are fixed
- Durations determined by flow
- Schedules constrained by durations

Multi commodity flow

	A	B	C	D	E	F	G	α_A	α_B	$\cdots$	α_G	β_A	$\cdots$
1	0..1	0..1	0..1	0..1				0..1	0..1	$\cdots$		0..1	$\cdots$
2	0..1	0..1	0..1	0..1				0..1	0..1	$\cdots$		0..1	$\cdots$
3		0..1	0..1		0..1	0..1			0..1	$\cdots$			$\cdots$
4		0..1	0..1	0..1		0..1			0..1	$\cdots$			$\cdots$
5				0..1	0..1		0..1			$\cdots$	0..1		$\cdots$
6					0..1	0..1				$\cdots$			$\cdots$
7						0..1	0..1			$\cdots$	0..1		$\cdots$
α_A	0..2											?	$\cdots$
α_B		0..4										?	$\cdots$
$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$						$\vdots$	$\ddots$
α_G							0..2					?	$\cdots$
β_A	0..2							?	?	$\cdots$	?		
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$		

MC-flow and scheduling constraints

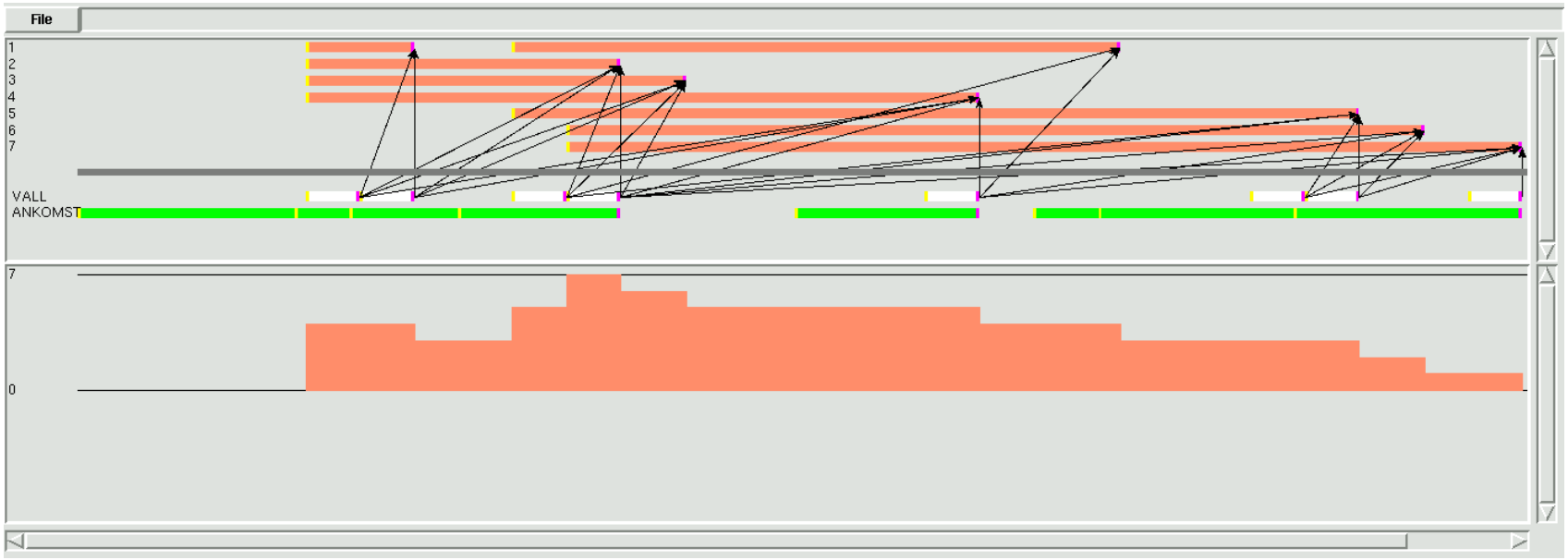
- Arrival and departure times for normal trains remain
- Sum of flow from any incoming train *i* into an outgoing train A and all potential temporary trains equal to flow into A in original flow
- For each potential temporary train and commodity (final destination train), the sum of inflows equals the outflow to the departing train corresponding to that commodity (unless some car is transported via several temporary trains)
- Scheduling constraints are updated with tasks for each potential temporary train
- Departure and duration for such tasks is constrained by transported commodities
- Flow through temporary trains is restricted by latest departure which is in turn constrained by transported commodities

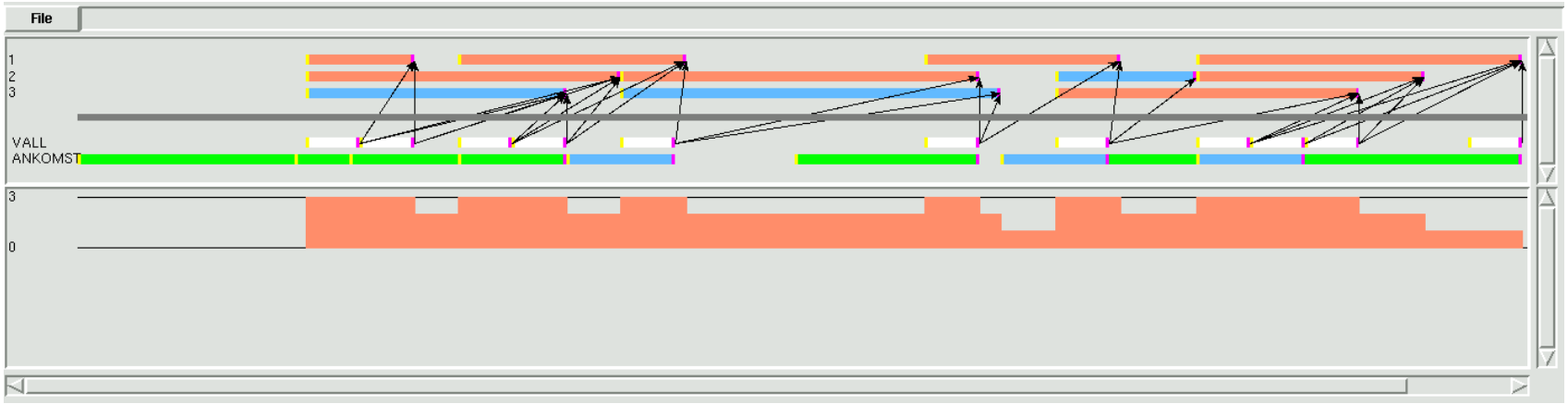
Outline

Trial runs

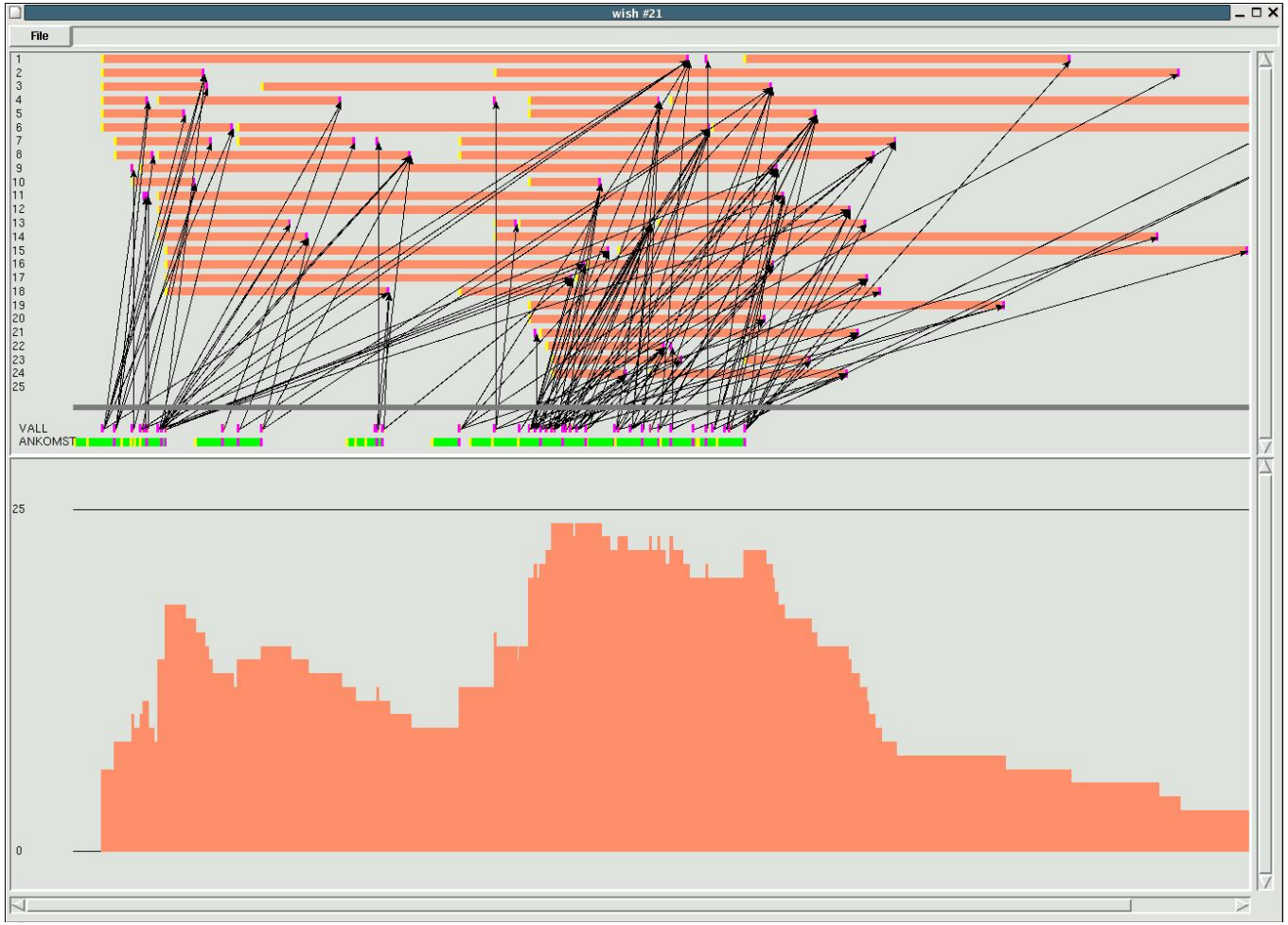
- Work in progress!
- Currently implemented in CP but may eventually be converted to IP
- For one temporary train and 8 outgoing trains, correct solutions are computed in the order or seconds
- For up to 3 temporary trains and on the order of 40 outgoing trains, correct solutions are *mostly* computed in the order of minutes
- The number of variables grows linearly with the number of temporary trains if we disallow flows between temporary trains but quadratically if we don't
- The model is full of nasty symmetries and hard to capture implications of the connections between scheduling and flow parts

With and without temporary trains

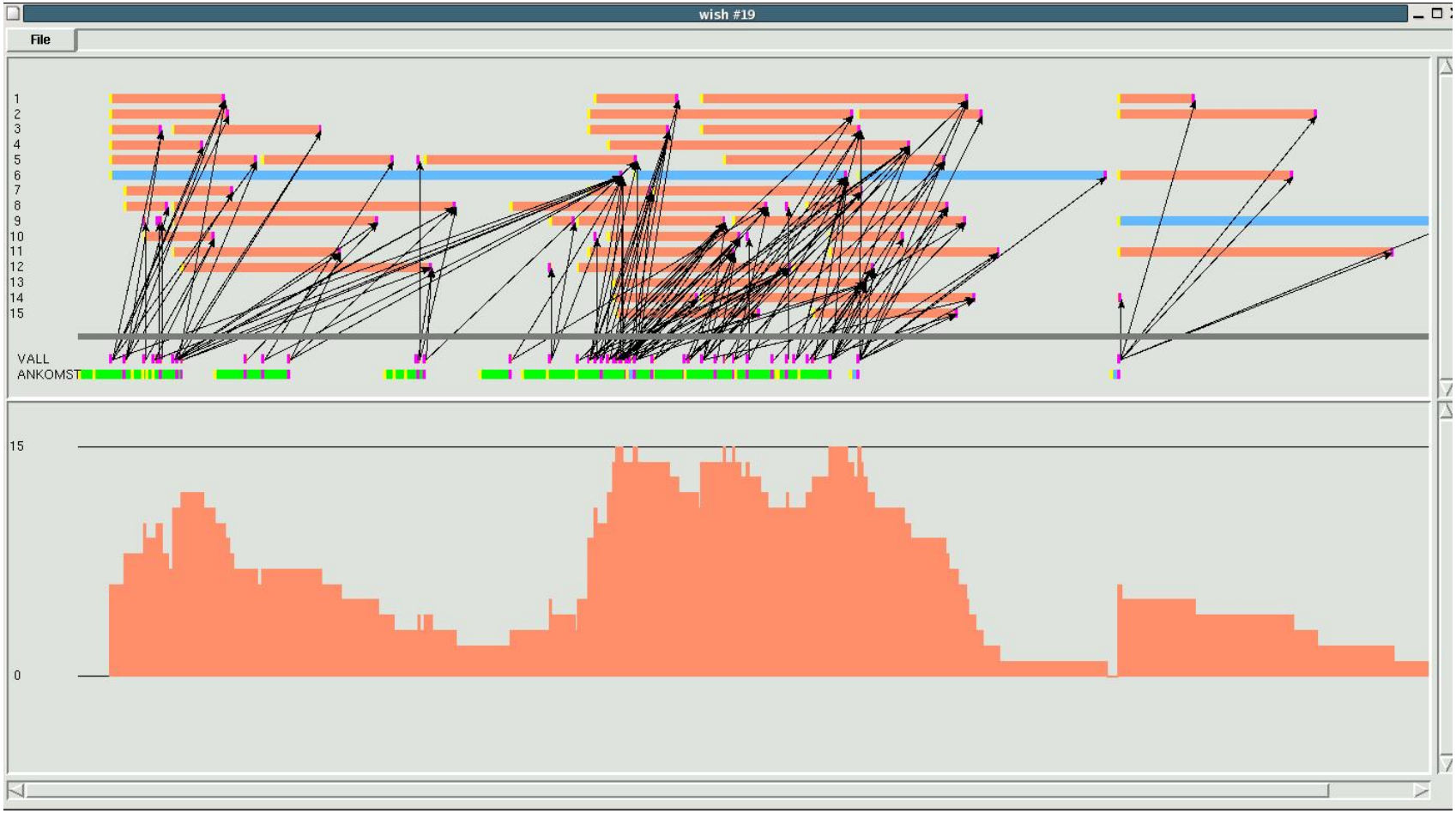




Sävenäs without temporary trains



Sävenäs with temporary trains



Outline

Lots to do!

- Seemingly straightforward problem turns out to be quite challenging
- The current model does not scale at all well
- Lots of symmetries to analyse and eliminate
- Search and cost model can be improved
- Choice of model and technique not at all clear
- Local search may turn out to be more practical
- Similarities with compiler register allocation problems
- Questions, comments, suggestions?