Home Page

Title Page

Contents

◀◀ ▶▶

◀ ▶

Go Back

Full Screen

Close

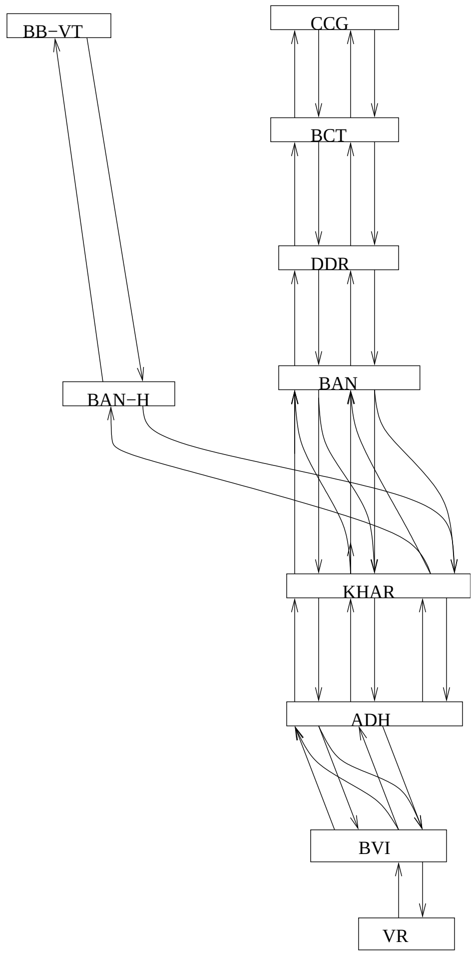
Quit

Railway Time-Tabling Effort

Milind Sohoni, Narayan Rangaraj and others
http://www.cse.iitb.ac.in/~sohoni

The WR Network

- 28 stations
- Over 200 track segments
- around 1000 services daily
- 67 rakes (physical trains)
- over 300 junctions/points



Objectives

Inputs

- The Physical Network
  stations, lines, platforms.
- Operational Norms
  Headway, turn-around times.
- Patterns of Operation
  such as `CCG-VR fast`
- Requirements
  Specific as well as aggregates

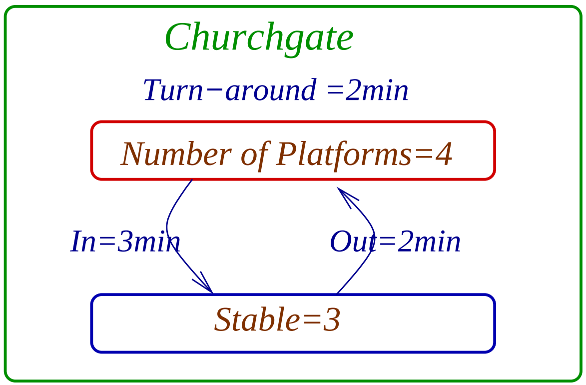
Outputs

- TimeTable
  detailed timings.
- Rake-Links
  alloting physical EMUs.
- Platform Charts.

The Network

Stations

- Name
- No. of Platforms
- No. of Stables
- Turn-around Time
- Push-In/Pull-Out Time



Lines-`unidirectional`

- Start/End Station
- Duration (time)
- Headway (time)
- Fork/Join List

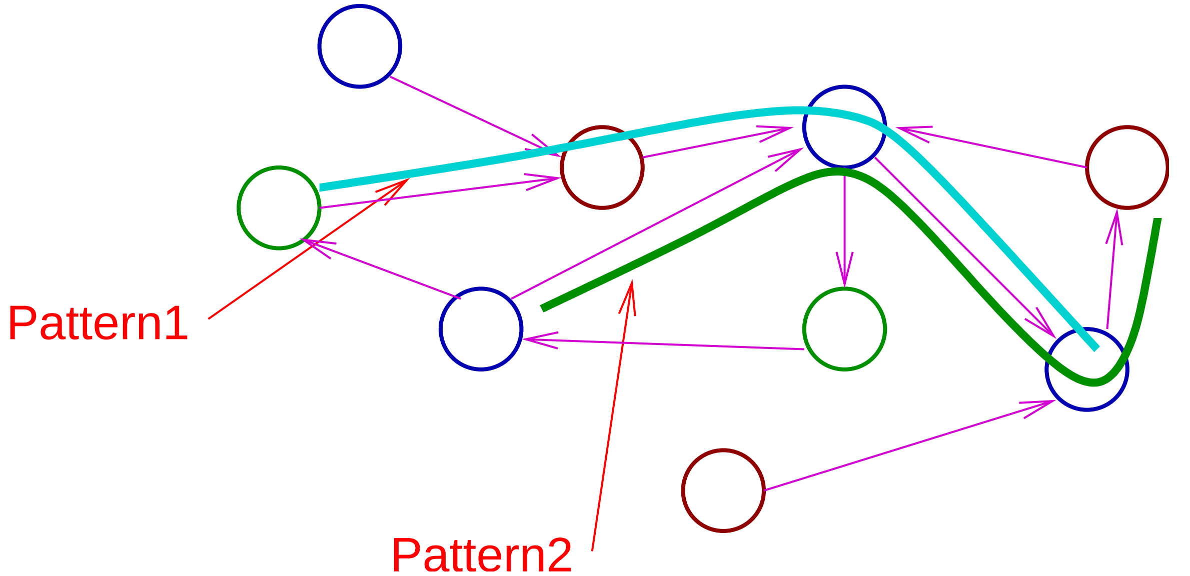
Pattern

This encapsulates a typical and repeating pattern of operation.

pattern-id		
9-CCG-BVI-Ft		
Station	Stoppage	Line
Churchgate	-	CCG-BCT-Thru
BombayCT	[1,1]	BCT-DDR-Thru
Dadar	[1,2]	DDR-BAN-Thru
Bandra	[1,1]	BAN-KHR-Thru
Khar	[0,0]	KHR-ADH-Thru
Andheri	[1,4]	ADH-BVI-Cross
Borivili	-	-

A Picture

Thus, pattern is a path with time prescribed flexibility in the network.



The *vptt*

The *vptt* is the input as well as the output. Fields are
(i) service-id and desired pattern
(ii) required start-time interval
(iii) actual start and end-times
(iv) rake-links

service-id pattern-id	`BVI-647` `9-CCG-BVI-Ft`		`PROP-3` `9-BVI-CCG-Su`		`ADH-751` `9-CCG-ADH-Sw`	
start-time	18:20	18:26	19:24	19:30	20:19	20:24
start	`CCG`	18:23	`BVI`	19:30	`CCG`	
end	`BVI`	19:24	`CCG`	20:27	`ADH`	
rake-link	`PROP-3`		`ADH-751`			

Home Page

Title Page

Contents

Page 8 of 19

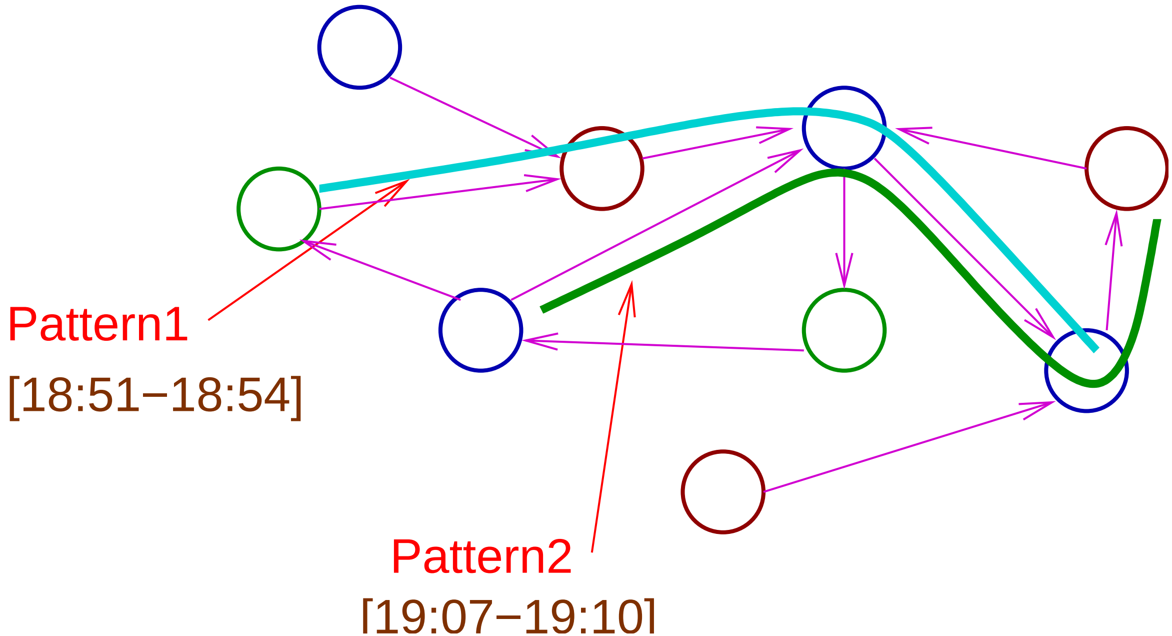
Go Back

Full Screen

Close

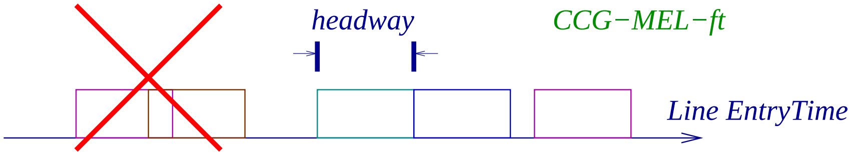
Quit

A Picture

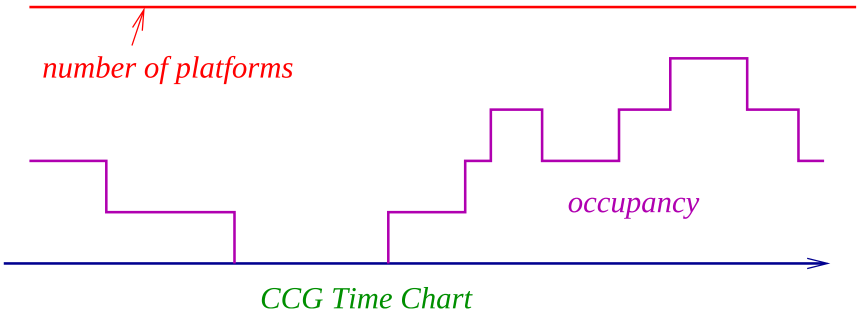


The Constraints

- line constraints-services using the same track must be headway apart.



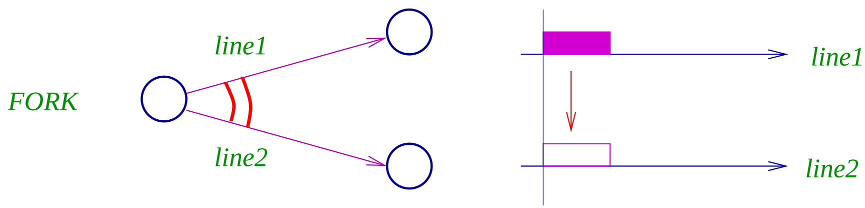
- platform constraints-these must be available for halts at stations.



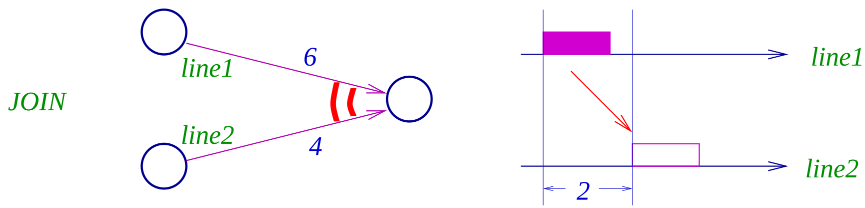
- continuity constraints-train departs a station and enters a line and vice-versa, and follows the pattern

Other Constraints

- Fork



- Join



Solvers

Manual Aids

- Check a TT
- Move services
- Order Services

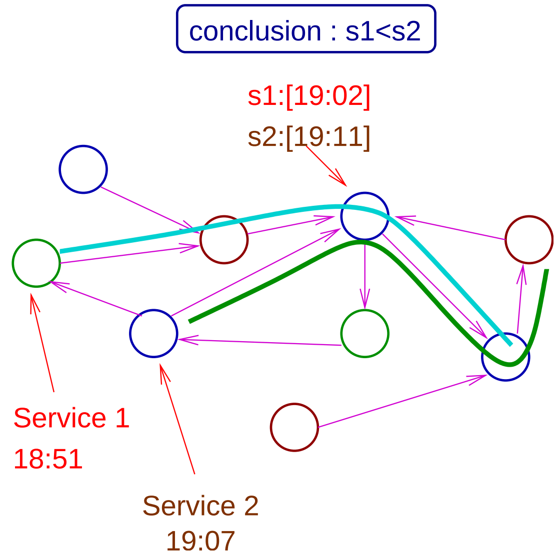
Automatic

- based on CHIP C++ Constraint Solver. Allows constraints to be posted on variables. Follows clever branch-and-bound
- Partial Order on services S by *time*
  Partition into clubs $S_1, S_2, \ldots, S_k$
- Solve S_i, S_{i+1} together. Freeze S_i and move to S_{i+1}, S_{i+2}

Compute-Instensive: Takes 50 minutes for half (UP) service set.

How to define S_i's?

- Organize the services in a temporal partial order.
- Pick bunch S_1 by peeling off the top few, then S_2 and so on.



- services s_i and s_j.
- depart-times t_i, t_j.
- patterns p and p'

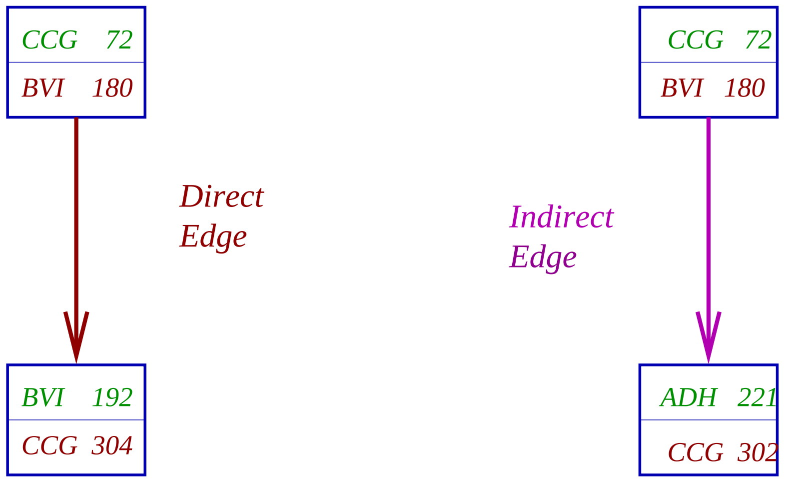
If $d_i - d_j \geq T(p, p')$
then $s_j < s_i$.

Rake-Linking

Once the services have been scheduled, they have to be provisioned: assign one of the 64 rakes available with WR.

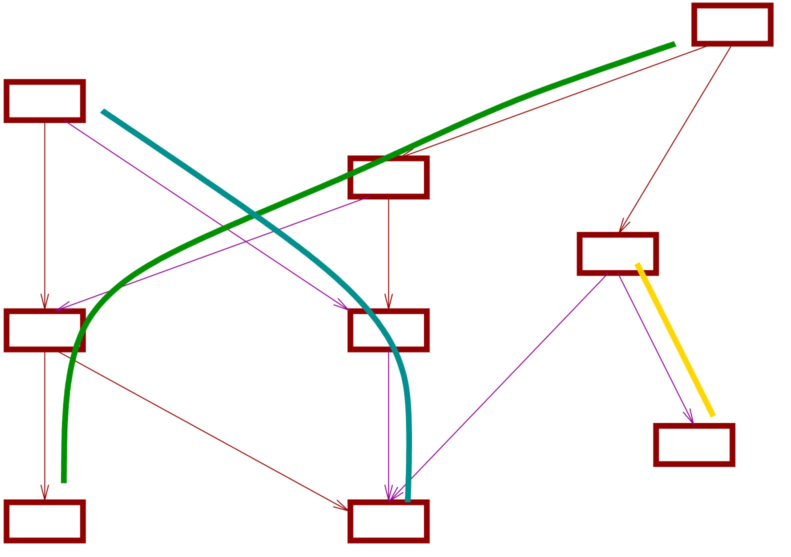
Step 1 Form the Service Graph.

- Vertices: Services
- Edges: Possible Successor



The Service Graph

Step 2 Compute Chain Decomposition.



- DAG
- Min-Cost-Flow and its variants
- Extremely Fast and provably optimal: 2 minutes

Platform Allocation

Inputs

- Service In-Out times
- Service Platform Preferences
- Set of Platforms

Service	In	Out	Platform
Rajdhani	18:56	18:57	4
Virar Local	18:54	19:05	1,2,3,4,5
Dahanu Shuttle	19:01	19:11	2,4

Output

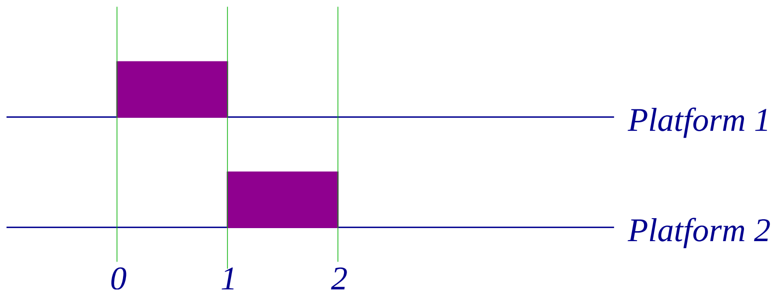
- Platform Allocation for each service
- Clash report if impossible

Undifferentiated and differentiated

Theorem: For the undifferentiated case, if at no point are there more than P services, then all services can be assigned platforms.

Thus a necessary condition is sufficient. However there is no such theorem in the differentiated case.

Rajdhani	0	1	1
Virar Local	0	2	1,2
Dahanu Shuttle	1	2	2



Home Page

Title Page

Contents

Page 17 of 19

Go Back

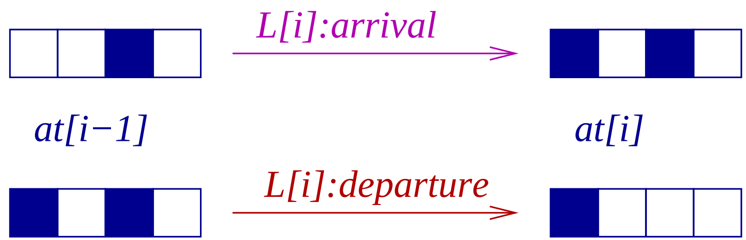
Full Screen

Close

Quit

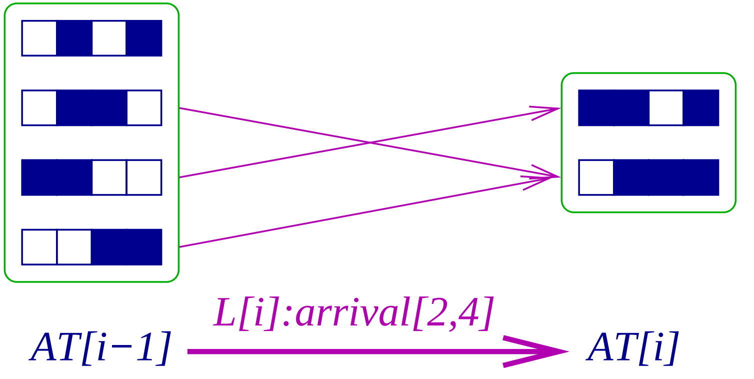
The Algorithm-undiff

- Let $A = \{a_1, \ldots, a_n\}$ be the set of arrival instants. Similarly, let D be the set of departure instants.
- Let $L = (l_1, l_2, \ldots)$ be the list $A \cup D$ sorted by time.
- Maintain Allotment table $at[i]$.



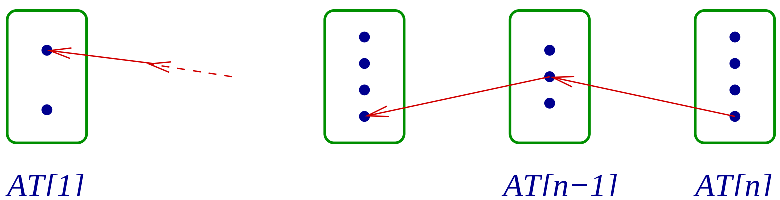
The Algorithm-diff

- Form L as before.
- Maintain $AT[i]$, the collection of all possible $at[i]$. This is essentially `Dynamic Programming`.



Algorithm continued

- If $AT[i]$ is empty for some i then declare `infeasible`.
- Otherwise, pick an $at \in AT[last]$ and trace back.



Complexity: Good for WR. Virar takes 20 seconds.