BTP Presentation

By Arpit Singh
111601031

Scheduling Problem

- **Resources** - Network topology, Stations, Tracks.
- **Train Movement** - Reference timetable (desired arrival and departure time of each train at each station)
- **Goal** - Assign track resources for each train for a fixed time period, such that they all complete their journeys without conflicts.
- Timetable may be **infeasible**.
    - Adjust arrival and departure times such that all rules are satisfied, while minimizing **Priority Weighted delay**.
- **Rescheduling** (online counterpart)
    - **Goal** : Recover from a disruption of timetable (due to delays or faults)

Overview of work

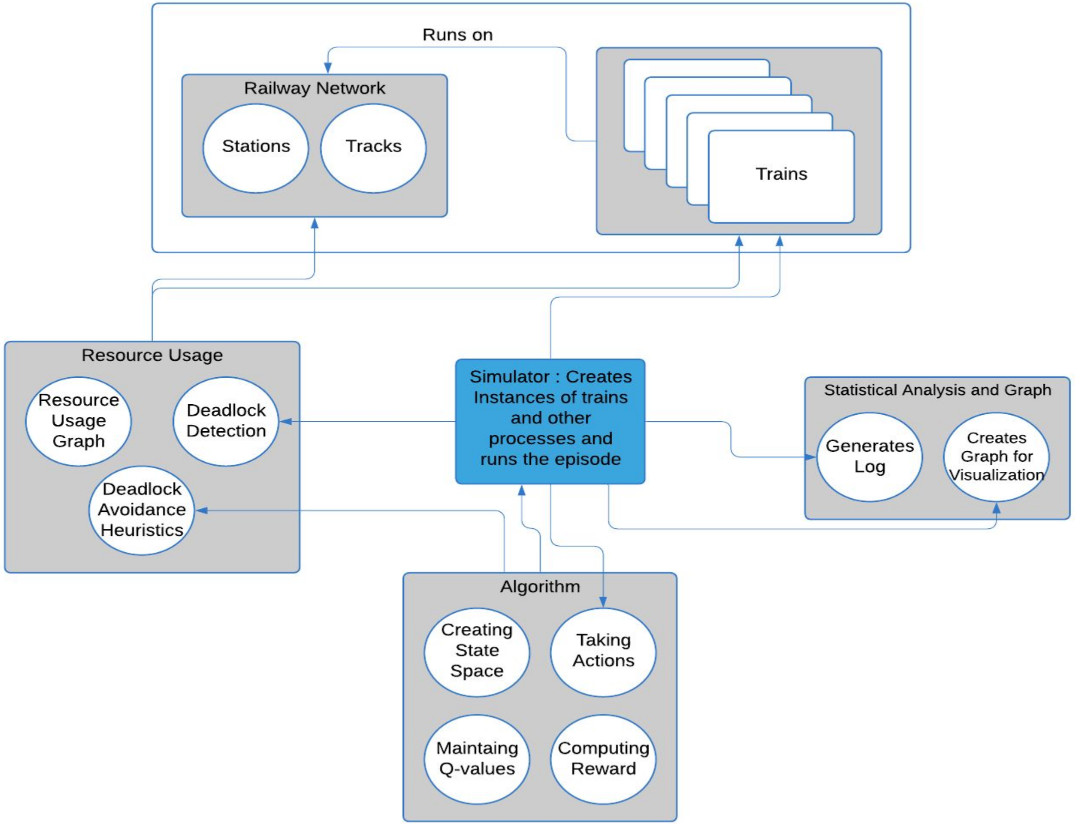
- Understood the problem statement of railway scheduling.
- Looked into prior work and proposed new approach.
- Designed the architecture of **simulator.**
- Implemented simulator.
- Implemented deadlock detection algorithm.
- Understood and implemented **deadlock avoidance heuristic.**

Railway Simulator - Requirements

- RL algorithm is driven by **discrete event simulator**
- Railway Simulator
  - Underlying railway network must be **flexible** and easy to tweak.
  - Train and railway network should be **independent.**
  - **Multiple trains** should run over network (Simultaneous processing of train)
  - Provides **logs** and **graphs** for visualization, analysis and debugging.
  - Keep tracks of the status of railway network and of each train (**Resource usage**).
  - Able to detect **deadlock** and handle it.
  - Algorithm should be attached as a pluggable component so multiple algorithms could be run and tested.

Key Ideas

- Create the underlying **static** railway network**.**
- Each train is a different **process**.
- **Multiple trains** run over same static railway network and **interact** with each other.
- Resource usage, graph generation, log generation, Algorithm are added as module to the system.
- A **central module simulator** controls all the processes in the system.`


Runs on
Railway Network
Stations
Tracks
Trains
Resource Usage
Resource Usage Graph
Deadlock Detection
Deadlock Avoidance Heuristics
Simulator : Creates Instances of trains and other processes and runs the episode
Statistical Analysis and Graph
Generates Log
Creates Graph for Visualization
Algorithm
Creating State Space
Taking Actions
Maintaing Q-values
Computing Reward

Simulator Working

Puts all the modules in place. First create the **railway network** and **simulation environment.**
Create different processes :

- Create **each train** as the process.
- **Choose action** : responsible for choosing action for trains.
- **Deadlock detection** (invokes after every predefined time)
- **Create Statistics** (checks the status of the network, terminates simulation if all trains are done)
- **Update Graph** (for visualization)

Actions on trains

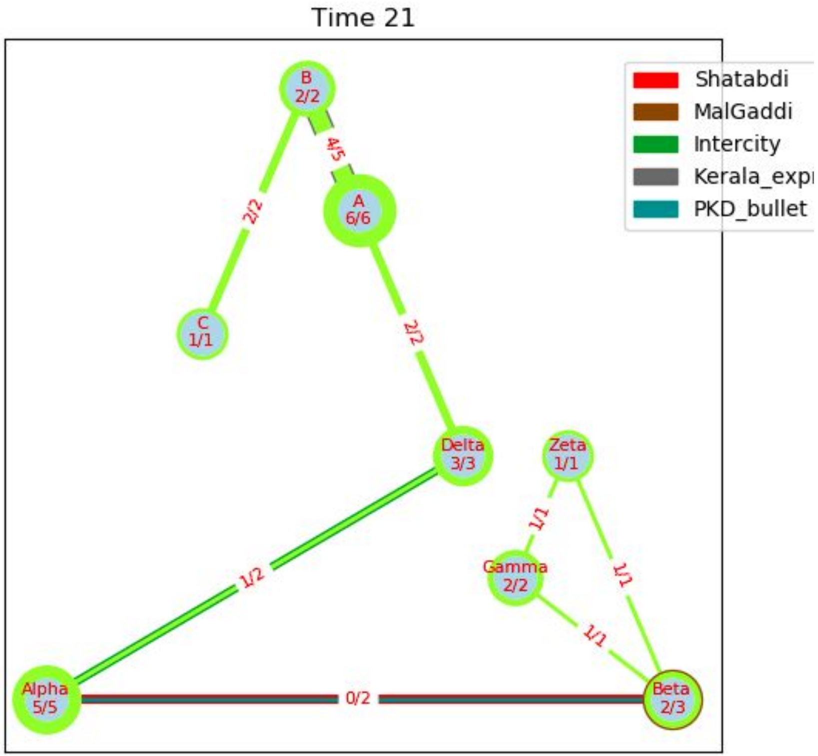
Two actions for train :

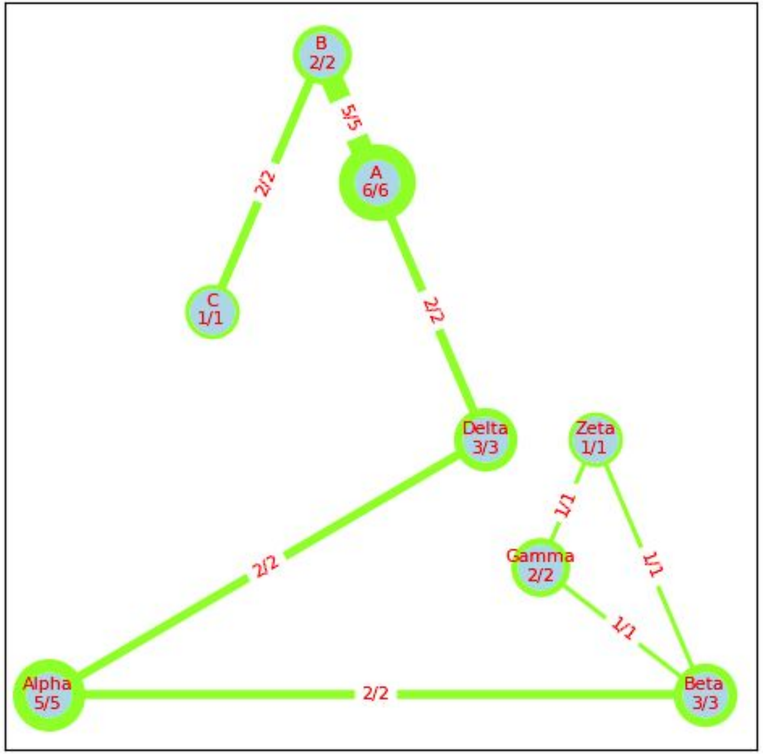
- **Move** to the next resource (station or track).
- **Wait** for predefined unit of time.
- Multiple trains can be waiting for action at the same time (so need ordering).

Time when trains need action

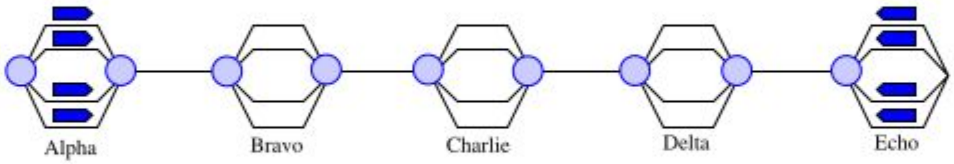
- When the train is standing at station and has to **depart to next track**.
- When the train is on track between two stations and ready to **arrive at the next station**.
- When the train is expected at the starting station.

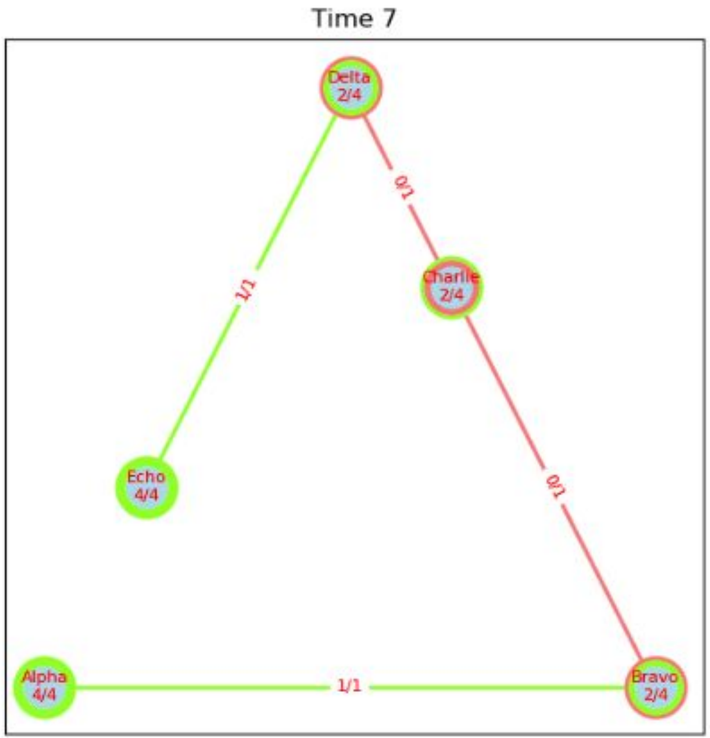
Graphs for visualization



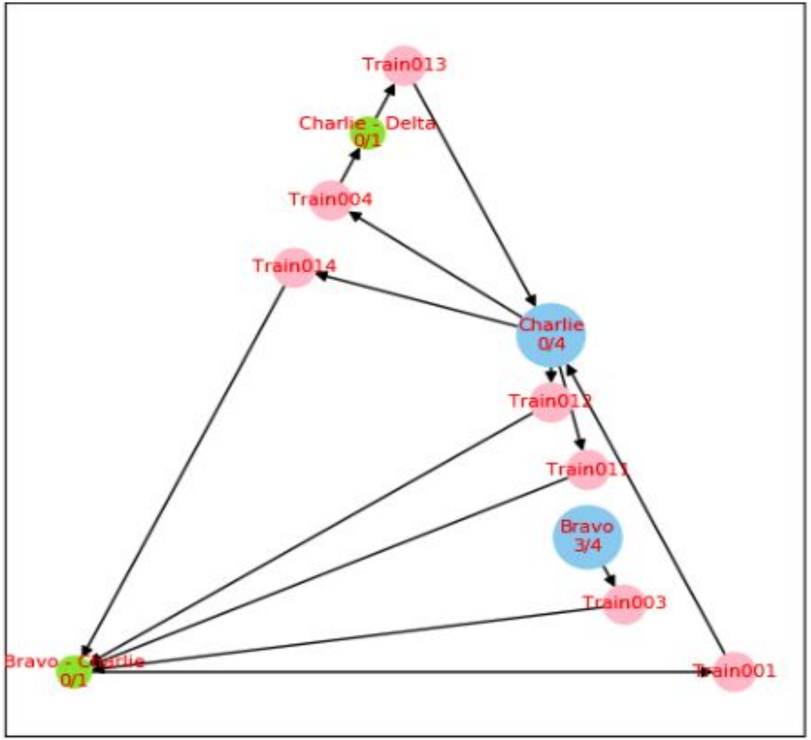


Toy Example


Alpha
Bravo
Charlie
Delta
Echo

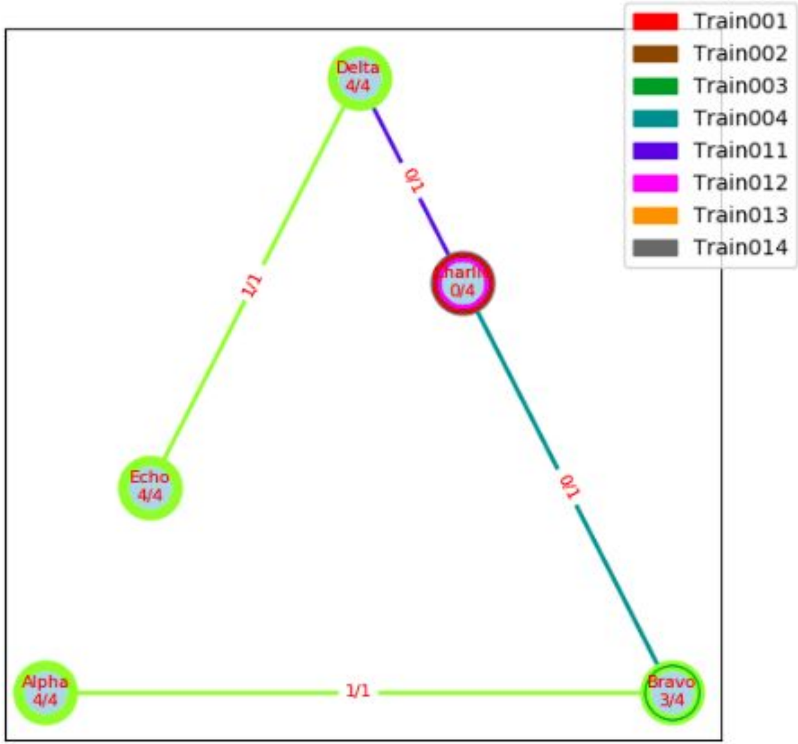

Time 7
Delta 2/4
Charlie 2/4
Echo 4/4
Alpha 4/4
Bravo 2/4
0/1
1/1
0/1
1/1

Resource Usage



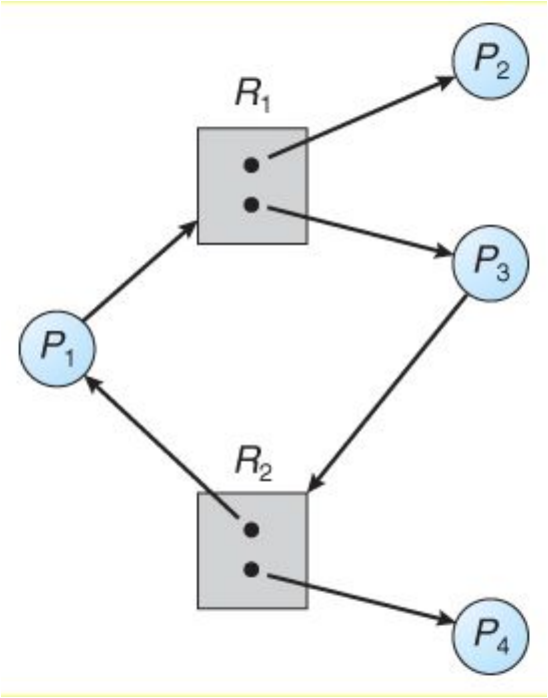
Deadlock

- Simulator encounters deadlock if the next chosen move is infeasible because
  - Train v finds all resources at the next node occupied by other trains, and
  - These other trains can only release their current resources if they move into the resource currently occupied by v.



Approaches

- Consider two adjacent resource, if all the **trains are moving towards each other** then it's a deadlock.
- Not always true.
- Find **cycle in resource usage graph.**
- Does not work if there are **multiple instances** of the resource (necessary but not sufficient).
- Final solution : **Banker's algorithm**



Deadlock Avoidance heuristic

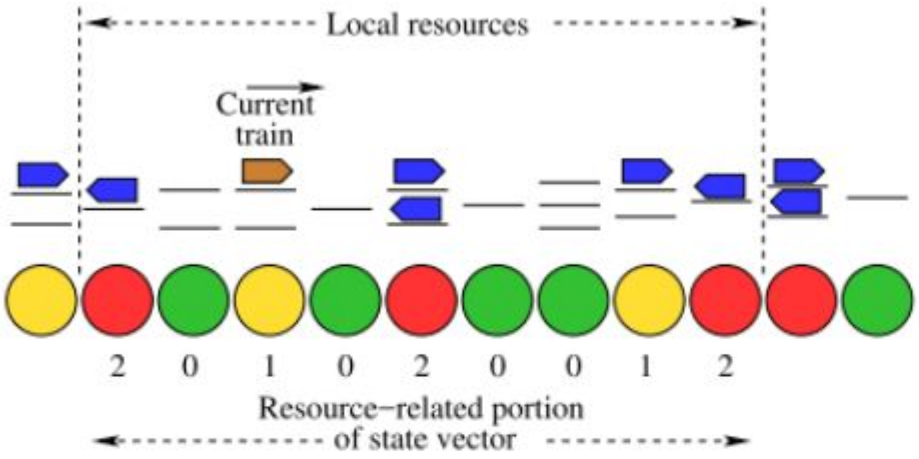
- Multiple trains need action at the same time.
- Which train to pick first.
- Use deadlock avoidance heuristic, that avoids deadlock but **not completely eliminate** it.
- Pick the train which is in the **most congested resource first**.
- The lower the number of free tracks in a resource, the higher the congestion.

State Space

- **Prior Work** : State space for each train, depends on local neighborhood.
- **Propose** : Taking the whole network into account and let the RL algo learn the important features.

Challenges

- Fead graph into neural network.
- **Variable size output** : depending on number of trains needing action.



Future Work

- Implement the RL algorithm with state space as in prior work.
- Check results , robustness , comparison to current algorithms.
- Use neural network as **function approximator** to consider bigger state space , hence improvising.
- Consider different state space.

Thank You