State Space Representation - Railway Simulator

CS4900 : UGRC Report

April 21, 2023

Hari Charan Korrapati
CS20B086
Faculty Mentor : Prof. Narayanaswamy N S
Indian Institute of Technology Madras

Abstract
This research is part of the Railway Simulation and Predction Project under professor Narayanaswamy, department of Computer Science, IIT Madras. The research project consists of three subsystems, Discrete event simulation, State space representation and Predicting actions, and Ontological representation of Railway domain.

This is the second part, which consists of the domain description and reprensenting the state space of the railway network, actions that can be performed on the network and predicting actions using basic search methods.

Keywords
State space representation, Automated Planning (Artificial Intelligence), Model Predictive Control, Indian Railways, Railway Delay Prediction, Event driven models vs Data driven models.

Contents

1 Domain definition

The domain definition and description of the Railway Network is given below. This includes the static, dynamic and control variales. At any instance (a discrete time interval), the state or configuration of the railway network can be expressed as a tuple of **(static varuiables, dynamic variables, control variables)**. This can be called as the state vector.

1.1 static variables

The properties of the railway network that do not change over time i.e their values remain constant throughout the entire simulation are represented as static variables. It incluedes:

train data:
Each train is uniquely numbered and identified with it. Trains are represented as $T_1, T_2 .... T_N$

Each train is a tuple, that consists of train no (the one given by indian railways, not the index), train index, train type (max speed), train length, train schedule information.

Mathematically, each train tuple is represented as
(T_{no}, T_{id}, type, v_{max}, T_{len}, schedule)

T_{no} - Integer
T_{id} - Integer
$type$ - $0, 1, 2$
v_{max} - $(0, 150)$ (in km/hr)
T_{length} - $(0, 2000)$ (in m)
schedule - vector

We are adding a train index field to uniquely identify each train, although we already have train no, trains that run over the span of more than 1 day, can have the same train no for different instances of the train. To avoid confusion we assign train-ids.

Type of train can be, super fast, express, or passenger. We assign numbering to this types. 0 is superfast, 1 is express and 2 is passenger. The train type enforces some constraints on the train, maxspeed, represented as v_{max}

schedule is a vector of tuples. Each entry in this vector is of the form, (station, arrival-time, departure-time) (S, t_a, t_d).
The service time in each station is $t_d - t_a$.

Mathematically, we represent stations with S and time with lowercase t and is given in seconds.

we are ignoring acceleration, braking and other capabilities of the train.

station data:
Each station is uniquely numbered and identified with it (like a primary key in ER diagram). Stations are represented as $S_1, S_2, ... S_N$

Each station is a tuple that consists of number of platforms and location. Mathematically, each stations is represented as

(S_{id}, P_{cnt}, latitute, longitude)

track data:
We view the railway network as a graph. A station in the railway network is a node and each track connecting any two stations is an edge. Each edge is considered to be direction independent. If there exists a bi-directional track in between stations then we have multiple edges between those nodes in our graph.

Each track is uniquely identified by a tuple (S_1, S_2) where S_1 and S_2 are the station ids of the corresponding stations in between this track exists.

Each track is uniformly divided into segments. Approximately 1km is the length of the segment. Each segment is uniquely identified. We have a standard for naming segments. Most trains are around 650m - 750m - loop length is usually less than 1km, so 1km seems reasonable. Good trains are much longer, like 1.5 - 2km, but we ignore them for now.

segment:
Each segment is given a segment id (using the standard segment naming protocol).

Segment naming standard:
$< S_1 > + < S_2 > + < d >$

Here, + represents string concatenation
S_1 and S_2 are the stations whose intermediate track encloses this segment.
d is the distance form S_1 in km

Each segment has two ids mapping to it. i.e each segment is given 2 unique ids. One from S_1 to S_2 and other from S_2 to S_1

if the distance between S_1 and S_2 is x km and it is d_1 km from S_1 and d_2 km from S_2
segment ids are $S_1S_2d_1$ and $S_2S_1d_2$
Intuitively, $x = d_1 + d_2$

1.2 dynamic variables

The following is a Temporal domain system, i.e the whole system evolves along with time. All the predicates and properties of the state space are dependent on time.

These are the properties of the network that are rapidly being updated as we keep running the simulation.

velocity (v), is represented in km/hr

location (d) distance from the source (or distance from a particular station in the train route) is expressed in km.

Each train is represented as a tuple,
train(T, seg, v, t)

T is the train and seg is the segment-id of the segment it is in at a time t and it is curently moving with a speed v km/hr

1.3 control variables

signal lights data:
Each signal is uniquely numbered. Each signal is a tuple of its location and the colour it is currently showing. Location is represented as the distance from either one of the stations of the track this signal is a part of.

Colour can be red/green/amber for each signal. We represent red as 0, green as 1, and amber as 2.

Mathematically, signal is represented as, (S_1, S_2, d, c, t)

d represents distance, c represents colour, t represents time.

2 Constraints

segment locking: Atmost one train can be present in each segment.

Platform locking: When a train is at a station, it blocks the platform for the duration of its service time and 1 or 2 minutes before and after its service time. It locks the platform, and no other train can take it.

Let S be the set of segments and T be the set of trains
$\forall t \in \{0, \infty\}, \forall seg \in S, \forall x \in T,$
$location(seg, x, t) \rightarrow \forall y \in T, x \neq y\ location(seg1, y, t),\ seg_1 \neq seg_2$

Initially, all the signals are set to green and all the segments are set to free.

$\forall S_1, S_2 \in S, \forall d,\ signal(S_1, S_2, d, green, 0)$ is true

$\forall seg, \in S, \forall d,\ free(seg, 0)$

For simplicity, we view the platforms as segments only. And each platform is given its id according to naming standard. We can do this because the segment locking protocols and platform locking protocols are very similar.

Loop lines are taken care by the simulator, when a train is waiting in a loop line we just assume it is waiting for the singal

3 Transition function

The transition function defines and constrains how the railway network can go from one state to another. The simulator takes care of the state transitions.

Train leaving and entering a segment:
Let's say that at a discrete time interval t, the train T is at location seg_1 and wants to leave the current segment and enter the next segment seg_2

First the train sends a segment clearance request signal requesting for occupying that segment at time t. In the next discrete time interval i.e at $t+1$ the controller sends back an accepting signal granting permission to access the segment.

Pre conditions:
location(T, seg_1, t)
next-segment(seg_1, seg_2)
free(seg_2, t)
signal(S_1, S_2, d, $green, t$)
seg-clearance-req(T, seg_2, t)
seg-clearance-acc(T, seg_2,$t+1$)

segmenting chaning phase:
This is the train entering the segment phase i.e it is between seg_1 and seg_2. We consider that this is almost instantaneous and takes only one discrete time interval to finish (few seconds). So, at $t+2$ train occupies the next segment so we have to lock it.

Post conditions:
free(seg_2, $t+2$) is set to 0
free(seg_1, $t+2$) is set to 1
signal(S_1, S_2, d, red)
signal($S_1, S_2, d-1, amber$)
signal($S_1, S_2, d-2, green$)

We use a request-signal and accept-signal protocol here like in networks or Operating systems concepts because at particular time t, more than one train can compete for a segment i.e multiple trains can send a segment access request at time t, so to avoid collisions we only grant access to one and others are not sent the accept signal at $t+1$. Next time instant is used to avoid race conditions.

rules for breaking ties:
we can define a set of rules to break ties for ex, rajhdhani or super-fast is preferred over passenger, or train with more delay is preffered over train with small delay etc.

Performing Action:
At $t+2$ just after entering the segment, we perform the action on the train if there is any.

$new-speed(T, seg, v, t+2)$

This will modify the speed inside the tuple T accordingly

Train cruising in a segment: Lets say that the train T entered the segment seg at t_1 (segment locking starts) and cruises in the segment till time t_2 i.e it is just about to leave the segment, the predicates are set as follows :

$\forall t \in (t_1, t_2)$ $location(T, seg, t)$ is true

We, assume constant speed inside a segment i.e if at the beggining of the segment, train has a speed v, it maintains the same speed throughout the segment. This is beacuse speed change is done through actions defined by the state space. The only way to modify speed is through signal posts at the begginning of each segment, so no changes in between.

So, time in segment (in seconds) $= 1000/v$, where v is in m/s
$t_2 = t_1 + 1000/v$

If the segment is a platform, then
$t_2 = t_1$+ service-time + 120
120 is 2 minutes, one minute is reserved for the train before and after entering the platform.

4 Actions

Actions define and constrain what all changes can be made to the state of the network by our controller (could be MPC or any other control algorithm). For now the only action we can perform is changing the speed by controlling signals.
train T that just entered segment seg with a speed v_0. Actions are defined as follows,
$new - speed(T, seg, v, t)$

5 Cost function

To implement a search and define an optimisation problem we should first define a cost function. cost is proportional to square of delay. This is better compared to linear addition of delays because a train arriving on time and another arriving more than an hour late is not ok. It is better to have both trains coming 30mins late (just a hypothetical case).

The cost needs to be calculated not only when a train is at the station but at each discrete interval. We define expected time of arrivals at each discrete and interval and calculate delays against it. This can be divided by evenly distributing time between 2 stations in its scedule.

For example, train T is sceduled at station A at 10:00 AM and B at 11:00AM (station that comes after A). if the distance is 30km, then ideally the train should be going at a constant speed of 30 kmph, we calculate delays at each point accordingly.

6 Predicting delays

This paper "A review of train delay prediction approaches" disccuses exhaustively about various prediction methods that can be used. It broadly classifies the algorithms into "event driven" vs "data driven".

"past repeats itself" - data driven

"capture and model the dependencies of railway network" - event driven (basically use a simulation)

Event driven approaches are hard to model and are also less accurate as for the size of the network they couldn't account for all the dependecies that affect delays. Also, most of these are stochastic, they provide probability distributions for train delays.

Railway networks constitute a heavily intertwined system as numerous train vehicles share a limited infrastructure of tracks. A single train running behind its schedule is likely to hinder other trains using the same tracks leading to cascading consequences. Due to this high degree of interdependency in railway operations, initial delays often propagate to other trains in the railway network

Algorithms that can be used to predict delays:
multi-variate linear regression (suitable for MPC)
decision trees
random forest
artificial nueral networks
support vector machine

problem of overfitting:
One paper stated that, purely data driven models have the risk of over fitting. Espicially, neural networks, they perform too well on the training data, but fail to do well when they have to predict on new data.

some of the factors that affect delays:
- current delay of each train
- delay at the origin
- weather conditions (breaking down into regions)
- day of the week
- delay of the previous days (for ex last 7 days)
- delay on the same date previous year (and neighbouring dates)
- delay of surrounding trains (running in similar routes and time)

Above mentioned are very few, we have to come up with a much more exhaustive list of factors and obtrain the corresponding data.

7 Data collection cleaning

Train and track route details - Indian railway's API (we can use python requests module), we have already scrapped some data.
https : *//railwayapi.com/*
https : *//data.gov.in/resource/indian−railways−time−table−trains−available−reservation*−01112017

This contains the train scedule of all trains (passenger only) as of 2018. we can also scrape from webistes like railyatri or trackmytrain etc to get the latest train data (2022 revised).

Train delays - Indian railways website
https : *//runningstatus.in/*
newline This website has delays of trains

weather data - OpenWeatherMap API
https : *//openweathermap.org/api*

This is one of the best sites to get past weather data pragmatically.

After obtaining all data, some cleaning has to be done, replacing null values with mean value etc. Also, we have to split the training set and test set properly. We have to normalize the data and do feature scaling. Also, we have to do data compression (dimensionality reduction - principle component analysis) as we will be having many features.

According to this paper "Prediction of Train Delay in Indian Railways through Machine Learning Techniques" and a few others, weather plays a significant role and is often ignored by many studies. Especially, in indian terrains, where rainfall occurs at different times of the year at different places and is highly unpredictable, it affects delays to a significant extent.

8 Acknowledgements

I am grateful to my guide, Prof. N.S. Narayanaswamy, for all his time and advice and for always being accessible to assist me. I would also like to express my gratitude to the Department of Computer Science and Engineering at IIT Madras for enabling undergraduate students like me to participate in research through the UGRC program. Many thanks to my research companions, Aditya Viraj and Yuvan, for constantly supporting me and always being there when I need help.

9 References