Framework for Applying Reinforcement Learning to Improve Railway Timetabling

MTP Phase - I

Submitted in partial fulfillment of the requirements
of the degree of

Master of Technology

by

Ramisetty Venkata Satwik
(Roll No. 193079009)

Supervisor:
Prof. Madhu N. Belur

Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
17^{th} October 2021

Abstract

Indian Railways carry an important role in economically transporting material and passengers across long distances. Currently, the Indian Railways has a high operating ratio. The operating ratio (in percent) is the cost to operate railway services compared to the income for the railway services of which means most of the revenue earned is lost in costs and maintenance. The major source of revenue of Indian Railways is freight transportation. Currently, the freight transportation timetable is less regular and affected by the busy network of passenger trains. Optimization of the network is possible for improving freight windows in the timetable and also to improve the average speed of the passenger trains.

The report gives an overview of the problem statement in terms of its application in Indian Railways and also emphasizes the motivation of this research and objectives. The principles of reinforcement learning and Markov Decision Process modeling of the Indian Railway network are explained. The algorithms used in constructing the environment-agent feedback loop framework are explained. The results and analysis of the framework have been explained. The scope for the future and continuation of this research work is also reported.

Contents

List of Figures

List of Abbreviations

RL	Reinforcement Learning
GQD	Golden Quadrilateral and Diagonals
MDP	Markov Decision Process
PIA	Policy Iteration Algorithm
SA	Simulated Annealing
IR	Indian Railway
RDSO	Research Designs and Standards Organisation
IITB	Indian Institute of Technology Bombay

Chapter 1

Introduction

1.1 Network overview

Indian Railways runs long-distance services that span over a large network, containing 17 zones (geographical regions) and over 70 divisions. The network consists of trains of mainly two categories, coaching trains, and freight trains. Coaching trains, generally known as passenger trains, further consists of subcategories based on the speeds (or priorities) of the trains. Broadly the trains run at speeds 90 kmph (kilometers per hour), 110 kmph, 120 kmph, 150 kmph. The coaching trains have a predefined order of priority based on which an overtake of one coaching train over another is decided. Usually, the high-speed long-distance trains are given higher priority and hence the lower-speed trains halt for the high-speed trains to overtake them. Freight trains, also known as goods trains, are classified based on the distance they cover during their traversal. The freight trains have the least priority in comparison to trains of other categories (including coaching trains). This is because the freight trains can wait for further time in their traversal when compared to coaching trains. An end-to-end freight runs typically 1500 km and halts at 4 to 5 stations for a crew change. The sectional freight trains run for lesser distances and have very few halts. A subsection of the Indian railway (IR) network consisting of the routes connecting Delhi (NDLS), Mumbai (MCT and CSTM), Chennai (MAS), and Howrah (HWH) is known as the golden quadrilateral. The interconnecting routes between non-adjacent cities form the diagonals of the quadrilateral. The GQD network carries heavy importance in the IR network as it economically connects the four major metros via high-speed services.

The importance of GQD network in IR is such that a timetabling is first completed on the GQD network. The remaining part of the IR network is handled in two ways. By considering

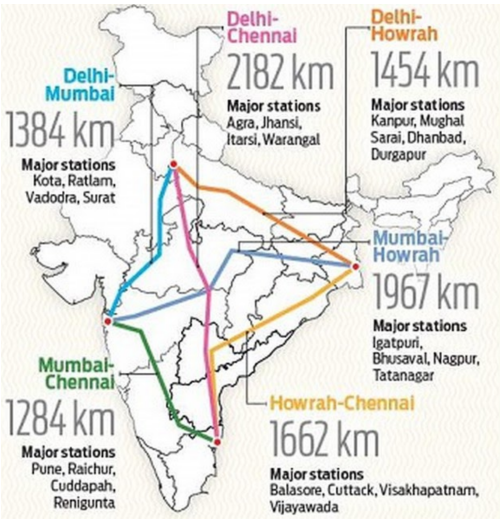

Figure 1.1: GQD network plotted on Indian map. (source: [1])

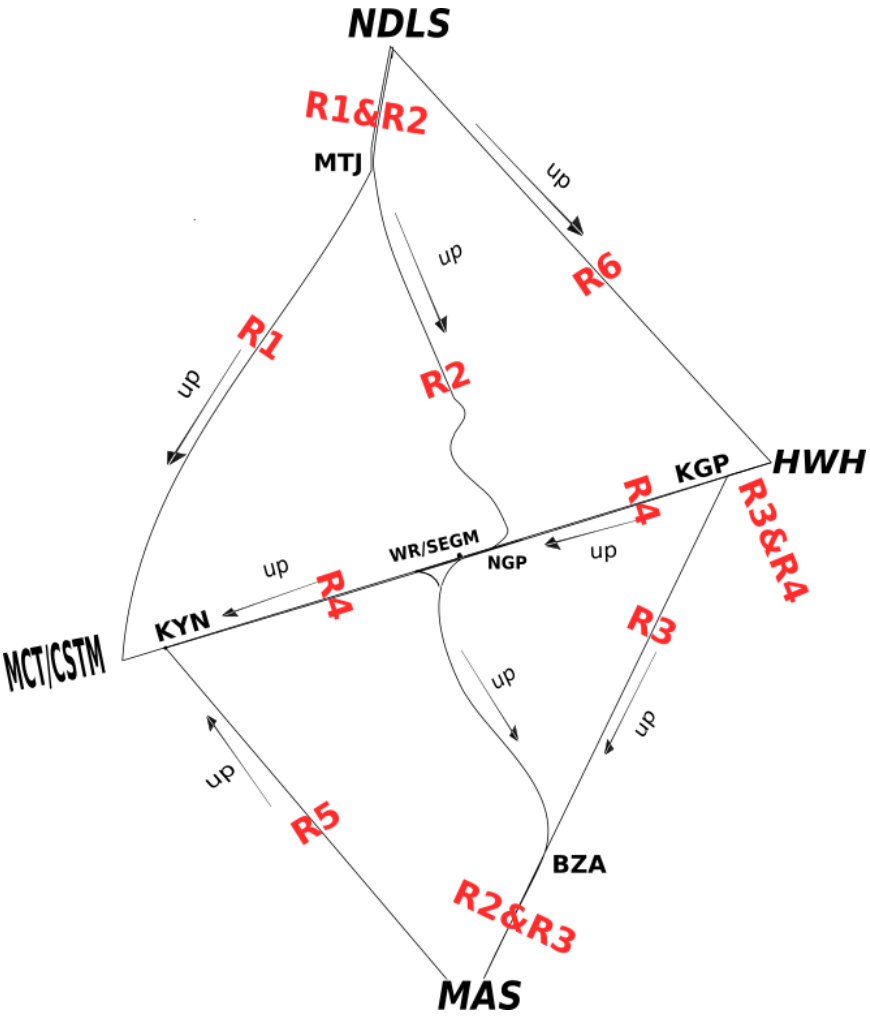

Figure 1.2: GQD network as graph, with junction and end stations labelled. (source: [2])

the non-GQD network as a constraint on the GQD timetable (for some high priority trains) and second, by computing paths as per the GQD timetable. This was acceptable because the GQD sub-network is generally acknowledged as the bottleneck part of the network [4]. Since the most constrained resource has a better say in the final resource allocation, hence beginning the timetabling from the most constrained resource such as GQD network can be considered as an optimal approach for tackling the IR timetabling problem. Optimizing a timetable requires a metric for computing the amount of delay for the trains in the timetable.

1.2 Performance metrics

A delay for a train is defined as the total time difference between scheduled arrival (or departure) and simulated arrival (or departure) time at all stations in the path of the train.

$$\text{Delay of } i^{th} \text{ train} = \sum_{Stations} \text{Scheduled Arrival time at station} - \text{Arrival time at station in simulation} \tag{1.1}$$

A delay of one train can have a severe snowball effect in the delay of multiple trains. Hence delays severely impact the performance of the IR network. To remedy the effect of stochastic delays, additional times are added to the traversal times of the trains. These additional times, known as allowances are consumed to compensate for the stochastic delays introduced in a zone of the train traversal. These additions of allowances to particular trains often have unintended effects on the rest of the network [5]. Therefore, it is an optimal choice to keep the amount of allowance limited. The amount of allowance needed is the higher if the interactions between trains in a timetable is high. This is because of the greater impact on the delay of other train by a train in such a timetable. Hence decoupling the trains to an extent will help in reducing the amount of allowance needed for each of the trains.

Another metric that contributes to the measurement of performance metrics of the timetable is the amount of utilization of the network that occurs in the timetable. This minimum expected utilization is especially useful in deciding the bottleneck stations in the network. For measuring the utilization of a station in 24 hours we look at the total amount of time the trains have occupied the station. Better timetables tend to balance this load and hence sudden spikes will not be

seen in the minimum expected utilization plots.

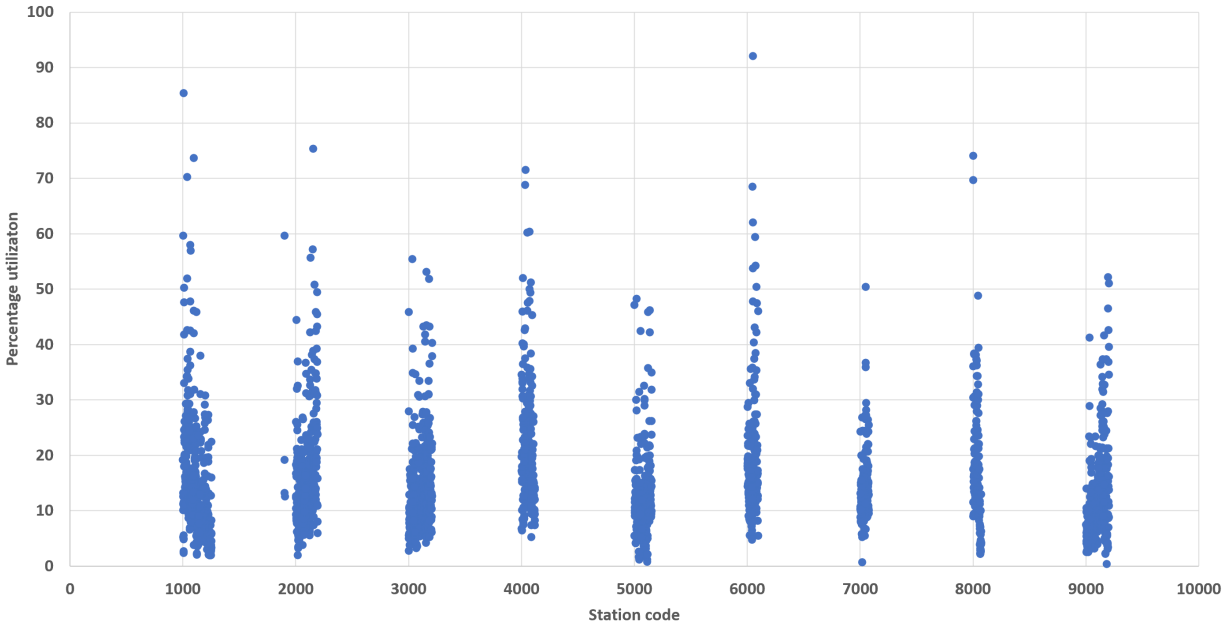

Figure 1.3: Percentage utilization at various stations. The stations are encoded with station codes that range from 1000 to 9000

Optimizing the GQD sub-network will result in a better throughput for coaching and freight trains. One method of measuring the amount of optimization is by using the average speeds of the trains in the GQD network. The higher the average speed of trains, the better the network timetable. The current research aims to design a systematic tool that would allow for eventual speed-up of the network.

1.3 Scope of research for extended problem statements

To generalize the problem statement to any network, we can study the key features that construct the problem statement. The main constructing ideas of the above network are:

1. A bounded or finite time-based **resource** for allocation

2. Presence of **heterogeneous sets of elements** or presence of **ranking among the categories of elements** (i.e. the allocation of resources to one element of a particular category is more preferred than another element of a different category)

3. All the elements considered must get a **unique resource allocation** at a given time step (i.e. no two elements can be mapped to exactly the same resource)

There are generally known in the art procedures for optimizing the above resource allocation problem. But the presence of the following additional constraints adds to the complexity of the problem statement.

1. A set of rules that must be adhered as an **element traverses from one node to another node**

2. A set of rules that must be adhered for **resource allocation between consecutive elements** in a network

In IR network, the available rail lines are finite resources that must be allocated to trains at a given time. The set of constraints that must be adhered to during the traversal of an element from node to node are the kinematic constraints such as the time for acceleration and deceleration of a train, a set of permanent speed restrictions that must not be breached during the traversal of the train, and the amount of time needed for siding a train near to a station, etc. Further, the set of constraints for resource allocation between adjacent elements in the network are the minimum headway times that have to be maintained between consecutive trains.

The solution of the current research can be extended to graph networks with the above properties. Hence with the appropriate changes, the current method can also be used to generate optimized schedules for other networks such as shipping, airways, and periodic roadway transportation systems such as bus schedules.

Further, in the field of communication, the current research can be used for resource allocation in time-division-multiplexing (TDM) systems. Here the net delay in the traversal time of the packets across the network can be minimized using the learning agent framework and an environment that models the connected systems appropriately.

Similarly, the current learning mechanism can be utilized for job scheduling or crew scheduling problems. The net overtime of the crew can be reduced using the current framework. Here the nodes of the graph correspond to the job locations and the set of constraints that must be adhered to from traversal between the nodes will be the amount of time required

for completion of a task. Some of the examples of additional constraints can be the time of rest between one job and another or cool-down time for a machine between tasks.

1.4 Motivation

The importance of optimizing the GQD network for increasing the number of passengers, and quantity of material transported in the IR network has been established in the section 1.1. Further, this optimization helps to improve the economy generated for the IR.

Operating ratio measures expenses as a percentage of revenue. The percentage of each rupee earned by the railways that are spent on its operations. As can be seen in the figure 1.4 with an operating ratio of over 90% railways has been struggling to generate a higher surplus [3].

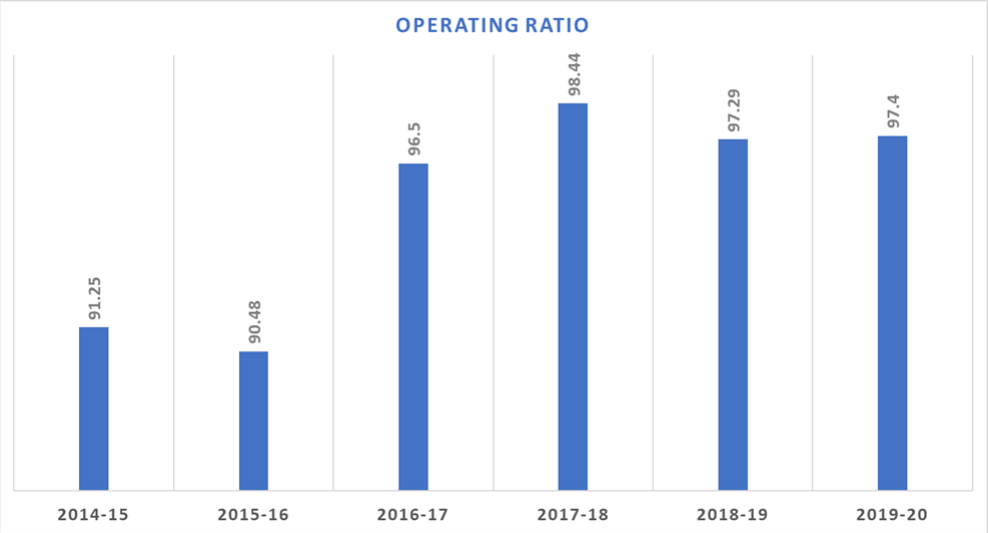

Figure 1.4: Indian Railways operating ratio between 2014 to 2020 as per data from [3]

Freight corridors are the main source of income for railways, hence dedicated freight paths will ensure lesser delays in freight transport and better revenue for railways. Figure 1.5 two important frieght corridors on the GQD network.

The present way of timetabling a train is for the zone where the train begins its journey to submit a preliminary timetable, i.e. the originating zone provides a departure time for the service from their end. This tentative plan is then used to timetable all subsequent zones suc-

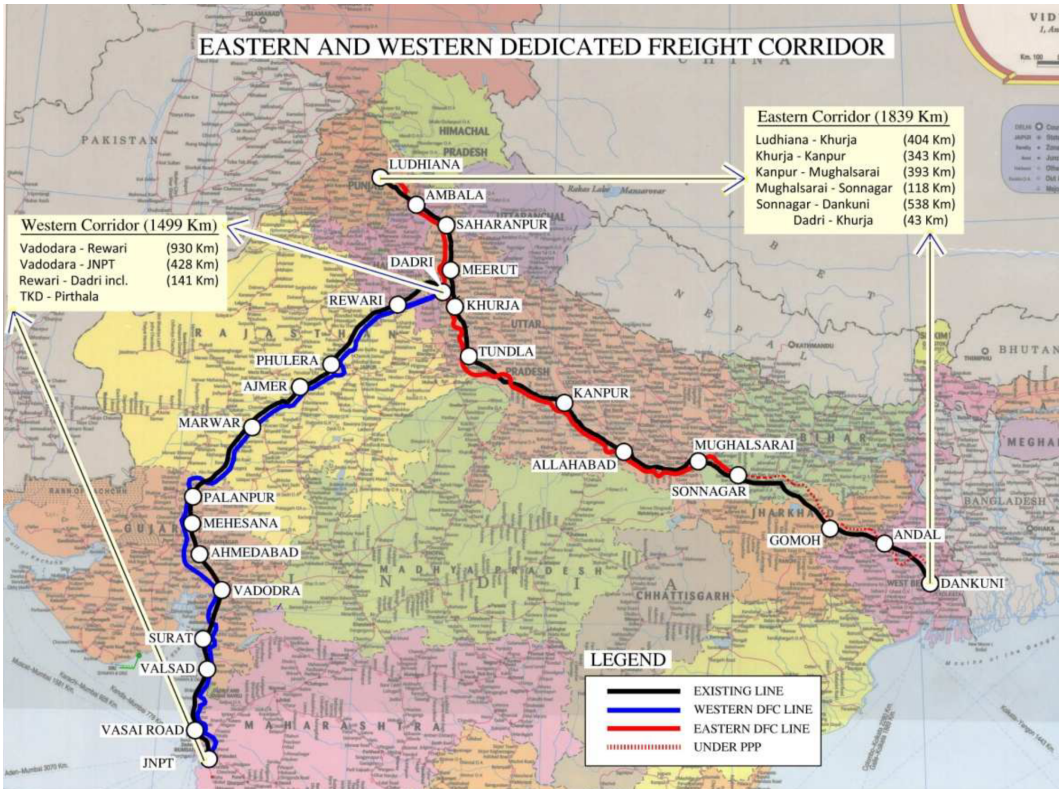

Figure 1.5: Eastern and Western Dedicated Freight Corridors Source: [1]

cessively. At the opposite end, the originating zone plans the return service. While scheduling the train in the other zones, there may be several constraints. The train's halts and speeds would necessitate resources at specific times, however, those resources may not be accessible due to other scheduled paths, maintenance blocks, and occasionally freight corridors planned on those portions at those times. Before settling on a final train schedule, the tentative schedule may go through multiple iterations.

This procedure necessitates cooperation across several zones and divisions, and it takes weeks to complete. Final coordination is generally done during an Indian Railways timetabling conference when all divisions/zones are represented and a preliminary end-to-end timetable for the train is provided. When done on a wide scale and several times, like in the case of the GQD network, this exercise might be tedious [2].

It is well known in the art that such timetable scheduling problems can be solved using mixed-integer linear programming. In theory, the railway scheduling problem can be expressed as an integer linear program. It is the NP-hardness of the problem [6] that proposes a challenge for computing a feasible challenge, thus rendering this method impractical in practise.

Recent techniques based on Reinforcement Learning (RL) have performed better for designing timetables from scratch using Reinforcement learning [4]. The use of RL to approximate the state action mapping has made the computation feasible to identify a solution for a proposed start timings for trains [7]. Hence, for the problem statements mentioned at the end of 1.1, because the computational effort is feasible, RL is a viable solution. However the additional constraints station from section 1.1 still remain unsolved in [4] and [7]. The main drawback with the method prescribed in [4] is that the model of environment does not account for many intricate details present in the actual IR network. The current thesis utilizes the RDSO simulator designed in IITB [8]. The advantage of the current simulator lies in the ability to model the IR network environment in a reasonably realistic form.

1.5 Research objectives

In this thesis, an agent is designed that can learn to improve an existing timetable through the iterative simulations. The main advantage of the current learning system over brute force solving of the timetable is to estimate the distributions of various set of actions and use the best possible action space to arrive at a improved timetable. Another aim of the thesis is the design an adequate cost metric that can improve the average speeds of coaching trains as well provide wide enough paths to freight trains.

Chapter 2

Introduction to Reinforcement Learning

2.1 Introduction

Artificial Intelligence (AI) is a simulated intelligence on programmable machines and tries to mimic the human brain. Machine Learning (ML) is a sub-field of AI, which concerns with "the question of how to develop software agents that improve automatically with experience" [1].

There are mainly three categories of ML

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

We require datasets (samples) with proper labelling for both supervised and unsupervised learning (targets). A domain expert initially provides these labelled samples. The goal is for the learning agent to be able to generalise its answers to situations that aren't part of the training set. Unsupervised learning identifies patterns in a labelled dataset to learn. Reinforcement Learning (RL) is a learning that is guided by a goal. An agent learns by interacting with an unknown environment in a trial and error manner. The agent receives feedback from the environment in the form of a reward (or punishment), which it then utilises to train itself and gain experience and information about the environment. RL problems are related to learning which is the best action to perform, situation-by-situation, in order to maximize the aggregated reward. The RL agent must learn a policy (i.e. a complete mapping between conditions and actions) by attempting actions without any input from a domain expert. Another important feature of an RL

problem is that the agent must choose between using its present knowledge of the environment (doing an action that has already been attempted in that condition) and exploring actions that have never been tried in that situation [9].

2.2 Reinforcement Learning in reference to Control systems

A control system's purpose is to determine the correct inputs (actions) into a system that will result in the intended system behaviour. The controller in a feedback control system employs state observations to improve performance and correct for random mistakes and disturbances. This information, as well as a model of the plant and its surroundings is utilized to create a controller that meets the system's requirements.

This notion is simple to understand, but it can be challenging to implement when the system is complex to model, extremely nonlinear, or has a huge state and action space. This is the case with the IR network which has over 2224 trains per day, and each train in a timetable needs separate corrective actions based on delays due to neighbor trains. As the action space gets larger the search space to determine the train to be modified increases as a nonlinear function which makes it harder to design a custom controller using techniques from control systems [4].

One easy way to address this problem is to divide it into smaller discrete portions that can be solved separately. To complete the overall state estimation, combine the solutions of the smaller discrete components. The estimated state and reference would be fed into the controller, which would most likely be made up of several nested control loops. The loops interact with one another, making design and tweaking difficult. It's also not easy to figure out the best method to structure these loops and divide up the difficulty [10].

2.3 Environment

Every RL problem requires the creation of an environment. Defining an environment entails laying down a set of rules, such as which actions an agent is allowed to take, what states the environment has, and what the rewards and punishments will be [11].

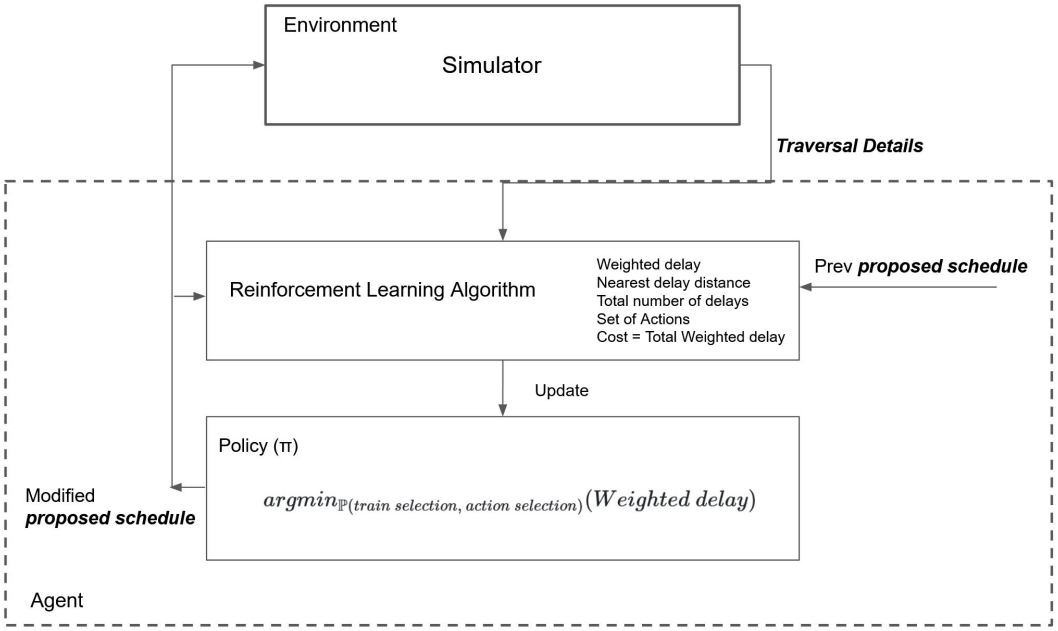

Figure 2.1: A block diagram showing a learning agent in feedback to the simulator

The world in which an agent or software program can interact or move is referred to as the environment. The environment's input is an agent's present state and action, while the environment's output is the next state and reward. The environment influences an agent's behaviours and outcomes (reward, resulting state).

In the current project the environment is the set of kinematic equations combined with safety constraints provided by IR. The environment along with the infrastructure can be simulated using a java program shown in figure 2.2 [8].

```
Reading Modified Traversal details file, NO interaction between trains ...
Done
Initial simulation ...
Creating unscheduled_arrivalTimes.csv
Running simulation with all trains together (with interaction) ...
/bin/bash internet.iitb.local.sh
cd simulator_input && \
  touch -m Delayfile.txt error.txt freight-details.xls Headway.xls Signal.xls TimeTableDetailed.xls TimeTableForComparison.xls Travers
alDetails.txt all_block_occupancy_details.txt
cp Maintenance-sub-3-days.txt Maintenance.txt && \
java -jar -Xss600m ~/jarfiles/Simulator-17th-Sept-16119f.jar non-interactive > ../logFiles/simulator.log 2> ../logFiles/simulator.erro
r.log && \
mv Delayfile.txt error.txt freight-details.xls Headway.xls Signal.xls TimeTableDetailed.xls TimeTableForComparison.xls TraversalDetail
s.txt all_block_occupancy_details.txt ../simulator_output_without/ && cd ..
/home/railwayint/satwik/1_NDLS-MMCT/simulator_output_without/TraversalDetails.txt /home/railwayint/satwik/1_NDLS-MMCT/simulator_input/
loop.txt /home/railwayint/satwik/1_NDLS-MMCT/simulator_input/station.txt
14680it [00:30, 487.51it/s]
Done
Reading Modified Traversal details file, WITH interaction between trains ...
Done
Generating priority lookup dictionary ...
Done
Generating train direction dictionary ...
```

Figure 2.2: The (java) simulator running in command line mode for simulating a proposed timetable

2.4 States

A state describes the current situation of the agent. Further, in the current thesis a state can refer to the state of the agent and the state an output from the agent. It is the learning and the outcome of the learning process that are the most important results for the thesis. Hence by default 'state of system' refers to the state of the output timetable from the agent. There can be a finite or infinite number of states. Terminal states are non-transitioned out states or states that bring the process to a close. A visit occurs when an agent goes from one state to another, and an episode occurs when numerous visits occur, such as from the start state to any terminal state.

A concise data extracted from a proposed timetable can be considered as the state of the system. An example for state of the system is shown in figure 2.3.

	minimum_tt(mins)	simulation_tt(mins)	traversal_time_delay(mins)	weighted_delay(mins)	_delay_distance(perc	est_delay_time(n	nber_of_delays
10002	942.55	942.55	0.00	0.00			0.00
10003	979.55	988.85	9.29	9.29	0.00	7.00	8.00
10005	937.55	995.95	58.39	58.39	0.00	13.00	34.00
10007	945.55	1024.19	78.64	78.64	0.00	20.00	50.00
10009	942.55	1028.46	85.91	85.91	0.00	27.00	49.00
10010	942.55	1033.06	90.51	90.51	0.00	34.00	59.00
10017	1052.85	1061.39	8.54	8.54	0.00	40.00	7.00
10018	1052.85	1075.09	22.24	22.24	0.00	47.00	19.00
10001	982.06	1089.44	107.38	53.69	0.00	54.00	53.00
10004	982.06	1097.27	115.21	57.60	0.00	61.00	58.00
10006	982.06	1104.14	122.09	61.04	0.00	67.00	71.00
10008	982.06	1110.00	127.95	63.97	0.00	74.00	74.00
10011	982.06	1115.78	133.72	66.86	0.00	81.00	75.00
10012	914.86	914.86	0.00	0.00			0.00
10013	648.16	648.16	0.00	0.00			0.00
10019	648.16	661.75	13.59	6.79	49.93	6.00	10.00
10020	982.06	1121.55	139.50	69.75	0.00	88.00	76.00
10014	290.22	457.78	167.56	55.85	75.02	76.00	3.00
10015	331.49	331.49	0.00	0.00			0.00
10016	1134.55	1337.25	202.70	67.57	39.72	25.00	4.00
12001	1502.20	1502.97	0.76	0.19	0.00	94.00	1.00
12002	1502.20	1516.83	14.63	3.66	0.00	101.00	13.00
12003	1502.20	1528.21	26.00	6.50	0.00	108.00	21.00
12004	1502.20	1541.13	38.93	9.73	0.00	115.00	27.00
12005	1502.20	1551.09	48.89	12.22	0.00	122.00	34.00
12006	1502.20	1561.87	59.67	14.92	0.00	129.00	37.00
12007	1502.20	1572.16	69.95	17.49	0.00	136.00	41.00
12008	1502.20	1582.76	80.55	20.14	0.00	143.00	44.00
12009	1502.20	1592.05	89.84	22.46	0.00	150.00	49.00

Figure 2.3: A state of the system containing 29 trains

In the current project the state is defined as the information containing the starting times, priority, acceleration, deceleration, halt stations and halt timings of the trains (also known as proposed schedule information) along with the delays with to ideal traversal time for each train. A proposed schedule is simulated and a next state is determined from the simulator.

The various features of a state are:

1. **Minimum Traversal Time**: It is the time required for a train to traverse from source (start) station to destination station, maintaining its halt pattern. The train is simulated alone without the presence of other trains to avoid any interaction based delay in the timetable.

2. **Simulated Traversal Time**: It is the time required for a train to traverse from source (start) station to destination station, maintaining its halt pattern. In addition, all the proposed trains are simulated together and hence the interaction (priority) based delays are present.

$$\text{Simulated Traversal Time} \geqslant \text{Minimum Traversal Time} \tag{2.1}$$

$$\text{Traversal Time Delay} = \text{Simulated Traversal Time} - \text{Minimum Traversal Time} \tag{2.2}$$

$$\text{Weighted Delay} = \frac{\text{Traversal Time Delay}}{\text{Priority of train}} \tag{2.3}$$

3. **Traversal Time Delay**: Equation 2.2

4. **Weighted Delay** Equation: 2.3

5. **Nearest Delay Distance**: It is the distance (of station) from the starting station of the train at which the delay as occurred. Based on the direction of train the distance is either with respect of zero milepost or the ending milepost of the route. This parameter can help in identifying trains that can be more optimized by postponing at the starting station itself. Since, if a train is delayed at the second/third station in its sequence it is better to delay the train at the starting station itself.

6. **Nearest Delay Time**: It the delay time that has occurred at the nearest delay distance station. Due to high interaction between the trains, the effect of this delay can compound on the subsequent trains during the journey of the train.

7. **Total Number of Delays**: It is the total number of delays that has occurred for a train during the simulation. This parameter tells us the number of overtakes the train has faced during its traversal. As the interaction between the trains reduce or if the paths of the trains are optimized, the over all number of delays faced by a train reduces.

A state of an agent is dependent on the agent. For policy iteration algorithm (PIA), an example for a state is shown in figure 2.4

2.5 Actions

A set of modifications that can be performed on a given state is known as action or collection of actions. In general, an agent is free to choose any action accessible in a given scenario.

In the project, four actions have been chosen.

1. **Postpone** the train with a time equal to the delay in traversal time of a train

$$\text{Proposed starting time} = \text{Previous starting time} + \text{Traversal Time Delay } \{2.2\} \quad (2.4)$$

2. **Postpone** the train with a time equal to the discounted delay in traversal time of a train

$$\text{Proposed starting time} = \text{Previous starting time} + \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \sum_{0\%}^{40\%} \text{Delay} \\ \sum_{40\%}^{80\%} \text{Delay} \\ \sum_{80\%}^{100\%} \text{Delay} \end{bmatrix} \quad (2.5)$$

   Here the weights w_1, w_2, w_3 are chosen as hyperparameters for the delay of train in the first 40% traversal distance of its route, 40% to 80% traversal distance of its route and above 80% traversal distance of its route

3. **Prepone** the train with a time equal to the delay in traversal time of a train

$$\text{Proposed starting time} = \text{Previous starting time} - \text{Traversal Time Delay } \{2.2\} \quad (2.6)$$

4. **Prepone** the train with a time equal to the discounted delay traversal time of a train

$$\text{Proposed starting time} = \text{Previous starting time} - \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \sum_{0\%}^{40\%} \text{Delay} \\ \sum_{40\%}^{80\%} \text{Delay} \\ \sum_{80\%}^{100\%} \text{Delay} \end{bmatrix} \quad (2.7)$$

Here the weights w_1, w_2, w_3 are chosen as hyperparameters for the delay of train in the first 40% traversal distance of its route, 40% to 80% traversal distance of its route and above 80% traversal distance of its route

Further the train on which the above four actions can be performed are chosen in three different methods

1. Choose a train that has **maximum amount of traversal time delay** (2.2)

$$\text{chosen train} = arg\ max_{train}(\sum_{stations} \text{traversal time delay}) \quad (2.8)$$

2. Choose a train that has **maximum amount of weighted delay** (2.3).

$$\text{chosen train} = arg\ max_{train}(\sum_{stations} \text{weighted delay}) \quad (2.9)$$

3. Choose a train with **highest number of stations with delay** for the train

$$\text{chosen train} = arg\ max_{train}(n(\text{delay})) \quad (2.10)$$

Hence in total 4x3 = 12 possible actions can be performed on a chosen timetable.

2.6 Markov Decision process

In a stochastic setting, the Markov Decision Process (MDP), a reinterpretation of Markov chains, is used for decision-making. MDP's purpose is to provide a mapping of the best ac-

tions for each condition of an environment. MDP is based on the Markovian property, which only considers current events and ignores previous knowledge. The prediction of the upcoming state is unaffected by previous states give that the objects or physics of the environment remain constant, and the laws do not change.

Intuitively it means that a state S, is sufficient to define the environment state and there is nothing else affecting how environment behaves [12].

The current method of simulating railway timetable can be considered as an MDP process for the following reasons.

1. Each simulation starts with a set of proposed starting times (proposed schedule data)
2. Once proposed schedule data is decided, then the corresponding delays associated with the state can be determined by performing simulation on the environment using the proposed schedule.
3. These delays are deterministic as the simulation only executes the timetable based on a set of precedence constraints and kinematic equations
4. Therefore the proposed schedule data has a one-one correspondence with the delays associated
5. Based on the obtained output an action is chosen, i.e. a train and an amount of delay or earlier dispatch is chosen
6. Hence once an proposed schedule data is considered and an action is chosen, the next state s_n does not require the information regarding any of the previous states and actions $((s_0,a_0),(s_1,a_1),(s_2,a_2),(s_3,a_3),\cdots,(s_{n-2},a_{n-2}))$ and hence the simulation-feedback process can be considered as a Markov decision process i.e.

$$\mathbb{P}(\text{next timetable}(s_n)|(s_0,a_0),(s_1,a_1),(s_2,a_2),\cdots) = \mathbb{P}((s_n)|(s_{n-1},a_{n-1})) \tag{2.11}$$

2.7 Transition model

It describes the environment's rules, dynamics, and physics. T(s,a,s') is a transition function that includes the current state, the chosen action, and the new state. It indicates the likelihood of shifting from state s (current state) to state s' (new state) by performing action a and receiving reward R. As a result, the transition model is unaffected by the past, i.e. states and actions, while being reliant on the present state, chosen action, and subsequent state (Markov property).

In the IR network, the actions that successfully reduced the amount of cost were selected with higher probability (when exploitation of existing data). Hence at a given time step the probabilities of the actions represent the transition function. Once an action is chosen, if the action is successful in reducing the net cost of the current state, the net count of the action is incremented, thereby increasing the probability of choosing the action.

$$T(s,a,s') = \mathbb{P}(\text{action}_1, \text{action}_2, \cdots, \text{action}_{12}) \tag{2.12}$$

Train_Action	0	1	2	4997	4998	4999
(0, 0)	1	1	2	504	504	504
(0, 1)	1	1	1	330	330	330
(0, 2)	1	1	1	125	125	125
(0, 3)	1	1	1	18	18	18
(1, 0)	1	1	1	29	29	29
(1, 1)	1	1	1	336	336	336
(1, 2)	1	1	1	36	36	36
(1, 3)	2	2	2	207	207	207
(2, 0)	1	1	1	731	731	732
(2, 1)	1	1	1	67	67	67
(2, 2)	1	1	1	137	137	137
(2, 3)	1	1	1	19	19	19

Figure 2.4: An example of the agent learning the best actions after 5000 iterations among the available 12 actions using PIA algorithm

Figure 2.4 is an example of the state of the agent changing with iterations. Each column in the figure 2.4 corresponds to a total probability of 1 with each actions having individual probabilities proportional to their counts. Hence each columns is a Transition model at the corresponding iteration step.

2.8 Reward [R(s)]

The Reward function is a feature that allows you to reward yourself for R(s) gives an agent a numerical value for being in a state after performing an action. Rewards indicate whether a state is beneficial or not, with more rewards given to agents moving in useful states and lesser rewards given to agents travelling in undesirable states. Agents receive feedback in the form of rewards, which can be favourable or bad. The nature of the reward, as well as the amount of it, has an impact on behaviour and policy (definition of policy will be given later). Minor adjustments can have a significant impact on agent behaviour. The cumulative future reward also called return is given as:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_{\infty} \tag{2.13}$$

In the IR network, the reward of 1 unit is considered if the action successfully decreases the cost of the resultant network. This reward increases the probability of selection of the actions that are effective in reducing the net delay in the timetable.

2.9 Policy

A policy guides an agent which action to choose in a given state. It is a mapping from a set of states to a corresponding set of actions. An optimal policy gives long term optimized reward which an agent could get in a lifetime. From policy, one may infer a plan, but a policy is different from a plan. A plan tells the sequence of actions from the start state to the destination state. A policy tells what to do in whatever situation you are in or which action is best an agent should take in a specific state. This is due to Markovian property which only depends on the present state. So a policy tells how an agent should act. An example for policy used for the 12 actions is shown in the figure 2.5. The train-action matrix (policy) is updated after each iteration of simulation.

Train_Action	0	1	2	3	4	5	6	7	8	9
(0, 0)	1	1	1	1	1	1	1	1	1	1
(0, 1)	1	1	1	1	1	1	1	1	2	2
(0, 2)	1	1	1	1	1	1	1	1	1	1
(0, 3)	1	1	1	1	1	1	1	1	1	1
(1, 0)	1	1	1	1	1	1	1	1	1	1
(1, 1)	1	1	1	1	1	1	1	1	1	1
(1, 2)	1	1	1	1	1	1	1	1	1	1
(1, 3)	1	1	2	2	2	2	2	3	3	4
(2, 0)	1	1	1	1	1	1	1	1	1	1
(2, 1)	1	1	1	1	1	1	1	1	1	1
(2, 2)	1	1	1	1	1	2	2	2	2	2
(2, 3)	1	1	1	1	1	1	1	1	1	1

Figure 2.5: A transition model with the counts of successful cost reduction actions in case of policy iteration algorithm. This is an example of how policy changes across iterations

Chapter 3

Development of Algorithms and Methodologies

Using the concepts of state, action, environment, cost, and policy, we can define an RL agent's purpose as finding the best policy for a given set of states to maximize long-term reward. The data flow diagram of the framework used for simulations is shown in figure 3.1

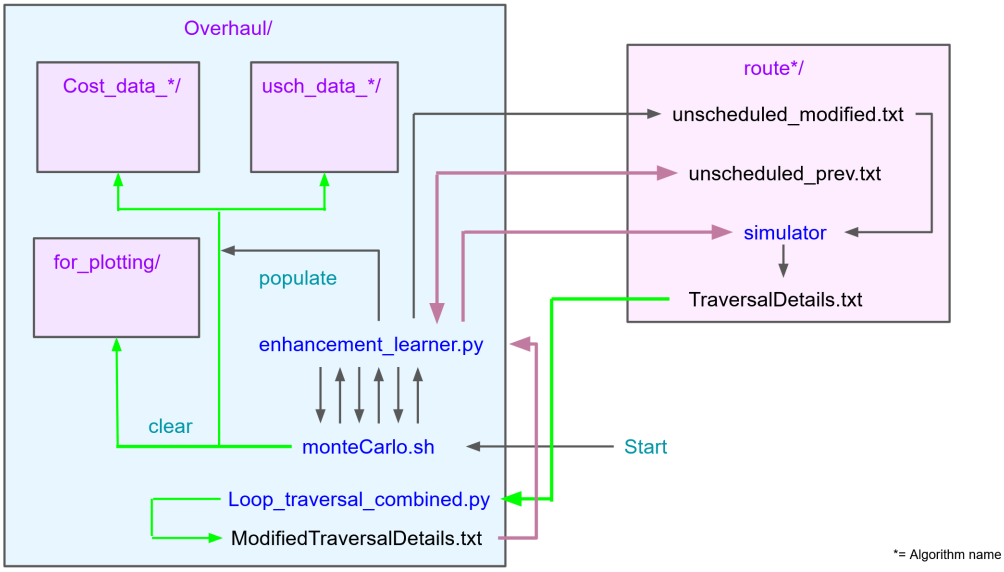

Figure 3.1: Framework used for testing various algorithms

3.1 Exploration vs Exploitation

One key issue in RL that does not evident in other machine learning algorithms is a delicate balance between exploitation and exploration. In exploitation mode, an agent chooses the best

action based on previously learned information, whereas in exploration mode, an agent may try random actions to improve its information in order to gain more reward. In the following algorithms at each iteration, a stochastic way of choosing actions is provided along with the best possible action. This ensures that the algorithm explores the action space for various trains, in spite of having enough data to exploit. This exploration vs exploitation is very much evident in the simulated annealing procedure where a metropolis criterion is used for updating a policy matrix.

3.2 Parallel execution of simulations

The simulations are conducted on an 8 core, 32 GB RAM machine. One iteration (epoch) of the entire simulation takes around 2 minutes 19 seconds. A typical simulation of one algorithm can have a minimum of 100 iterations. Hence the need for computational optimization is important. Much of the simulations can be conducted faster if they are run on parallel threads in the machine. To enable the faster execution, make files are used for making several copies of the original directory and running the simulations for each of the directories in parallel as shown in figure 3.2. With the parallel execution 4 independent simulations, each having 100 epochs can be executed within 3 hrs 27 minutes.

3.3 Policy Iteration Algorithm

In the policy iteration, our target is learning of an optimal policy by iterative use of the Bellman equation. The policy algorithm has three steps. First, select any random policy and perform policy evaluation. In the second step, improve the policy by using the value function and finally repeat first two steps until convergence. The pseudo-code for Policy Iteration Algorithm:.

PIA manipulates the policy directly, instead of finding it indirectly through optimal V(s) or Q(s,a) . It is a method in RL that assists in learning the optimal policy.

For all the algorithms the initial proposed schedule considered is that all the trains depart their starting station at 1 am in the morning. Since the simulator generates feasible and conflict

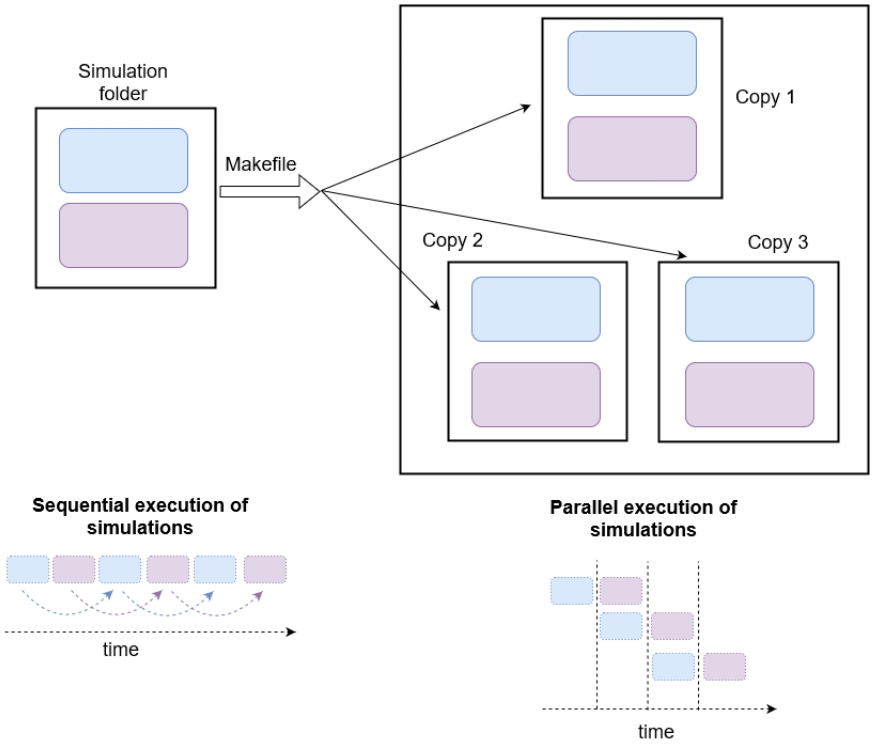

Figure 3.2: Parallel execution of simulations

1. Initialization

$V(s) \leftarrow U[0,1]$;

2. Policy evaluation

while $N \neq 0$ **do**

  if $U(0,1) < \alpha$ **then**

    $a \leftarrow Rand\pi(s)$;

  else

    $a \leftarrow max(\mathbb{P}(\pi(s)))$;

  end

  $N \leftarrow N - 1$;

  if $\Delta C < 0$ **then**

    $V(s) \leftarrow a + 1$;

  end

end

Algorithm 1: Policy Iteration Algorithm

free timetable, the trains depart with a delay and the output of the initial simulation is shown in figure 3.7. Here the the X axis of the graph is time in minutes and Y axis denotes a mile post distance (distance of a station w.r.t. to a zero milepost)

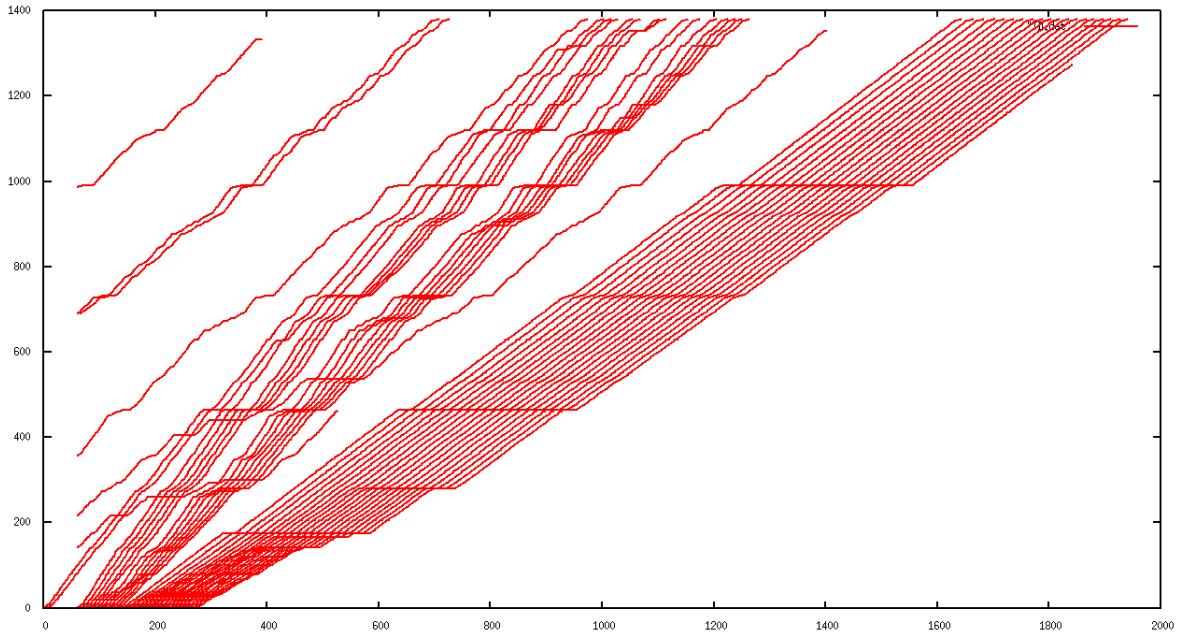

Figure 3.3: Initial proposed schedule

An intermediate state of simulation after 30 iterations of PIA is shown in figure 3.4. A final state of simulation after 100 iterations of PIA is shown in figure 3.4

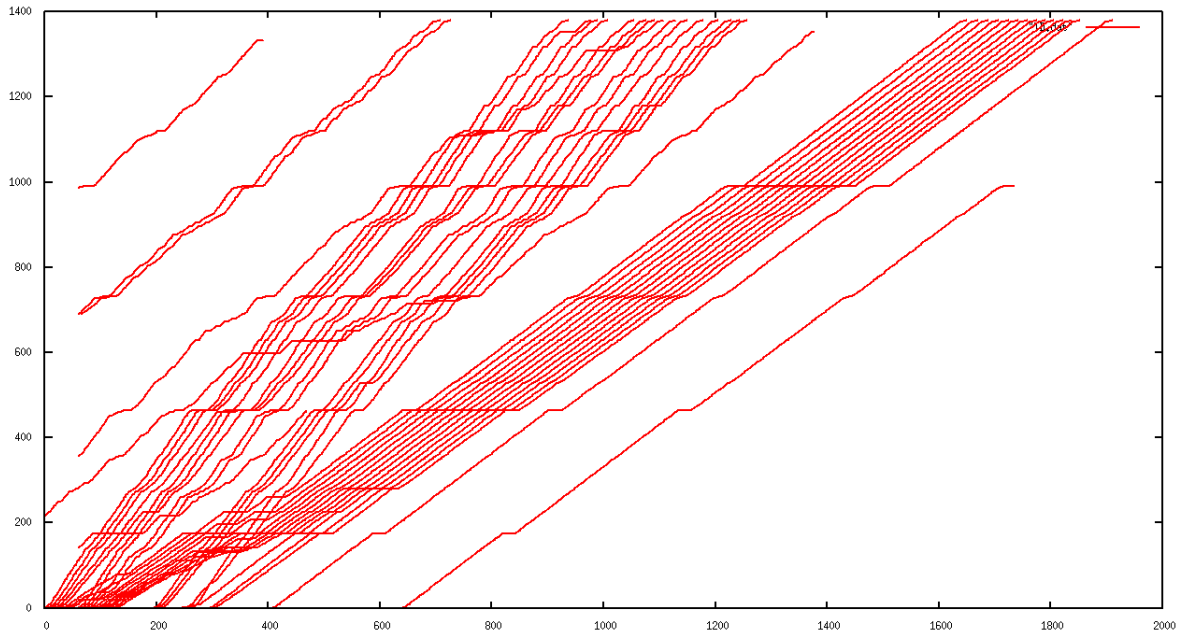

Figure 3.4: Distance vs time graph after 30 iterations of PIA

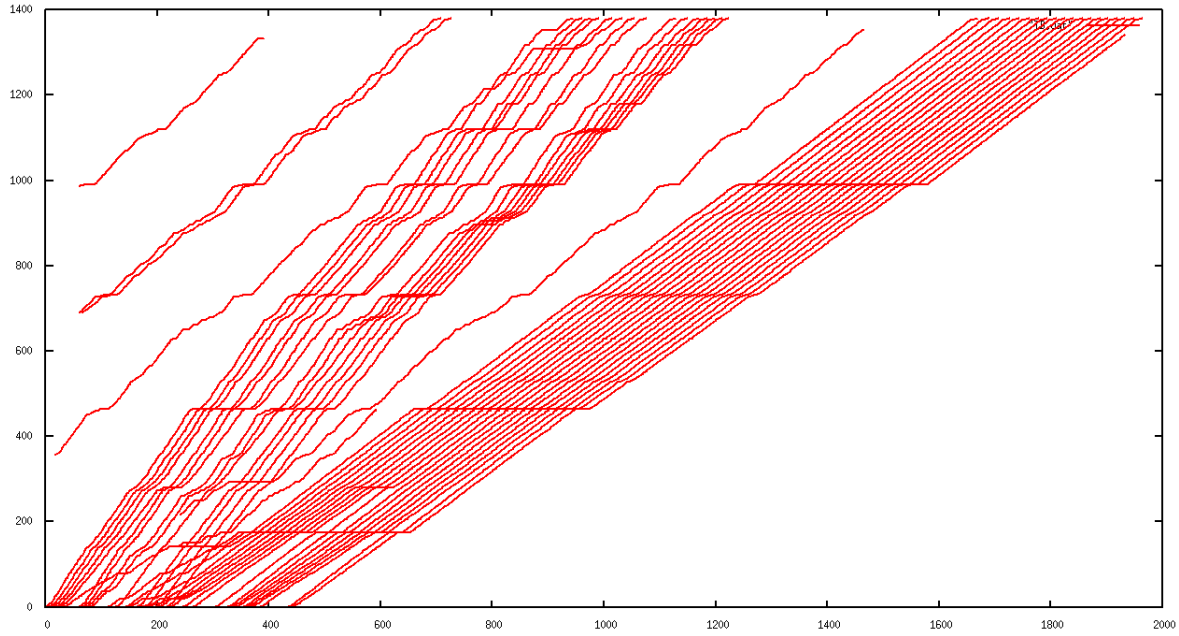

Figure 3.5: Simulation of train vs distance after 95 iterations of PIA

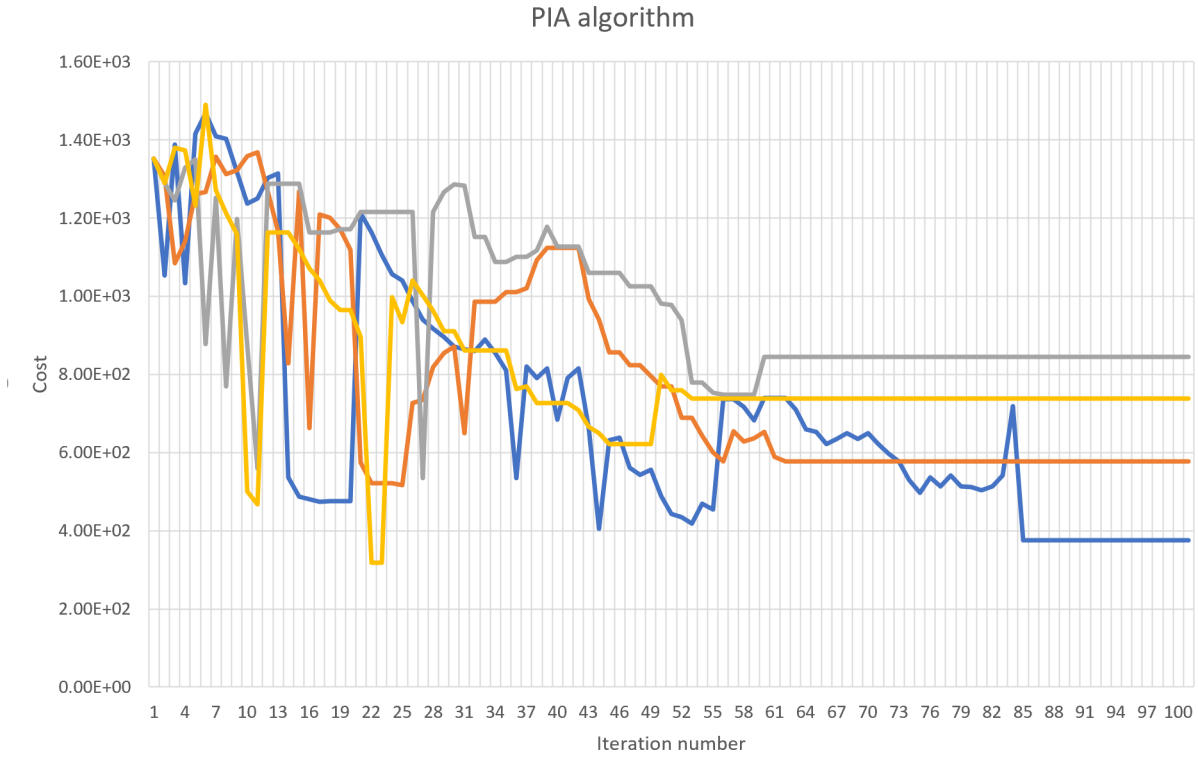

Figure 3.6: Comparison of 4 different PIA simulations

3.4 Simulated Annealing

The simulated annealing (SA) algorithm is an optimization technique that has been successfully used for solving a wide range of combinatorial optimization problems. The SA algorithm is an

optimization procedure based on the behavior of condensed matter at low temperatures that mirrors the annealing process that takes place in nature. An important characteristic of the SA algorithm is that it does not require specialist knowledge about how to solve a particular problem. This makes the algorithm generic in the sense that it can be used in a variety of optimization problems without changing the basic structure of computations. The procedure employs methods that originated from statistical mechanics to find global minima of systems with very large degrees of freedom. The major advantage of SA algorithm over the pure local search method is the ability to avoid becoming trapped in a local solution.

The algorithm is very similar to PIA except for the updating step.

1. Initialization

$V(s) \leftarrow U[0,1]$;

$T \leftarrow T_0$;

2. Policy evaluation

while $N \neq 0$ **do**

  $T \leftarrow T/N$;

  if $U(0,1) < \alpha$ **then**

    $a \leftarrow Rand\pi(s)$;

  else

    $a \leftarrow max(\mathbb{P}(\pi(s)))$;

  end

  $N \leftarrow N-1$;

  $D \leftarrow$ current cost $-$ previous cost;

  $M \leftarrow e^{(\frac{-D}{CT})}$;

  if $\Delta C < 0$ ***or*** $Rand < M$ **then**

    $V(s) \leftarrow a+1$;

  end

end

Algorithm 2: Simulated Annealing

An intermediate state of simulation after 30 iterations of SA is shown in figure 3.8

A final state of simulation after 100 iterations of SA is shown in figure 3.8

Here the SA does not update the result until either the cost has decreases or the metropolis

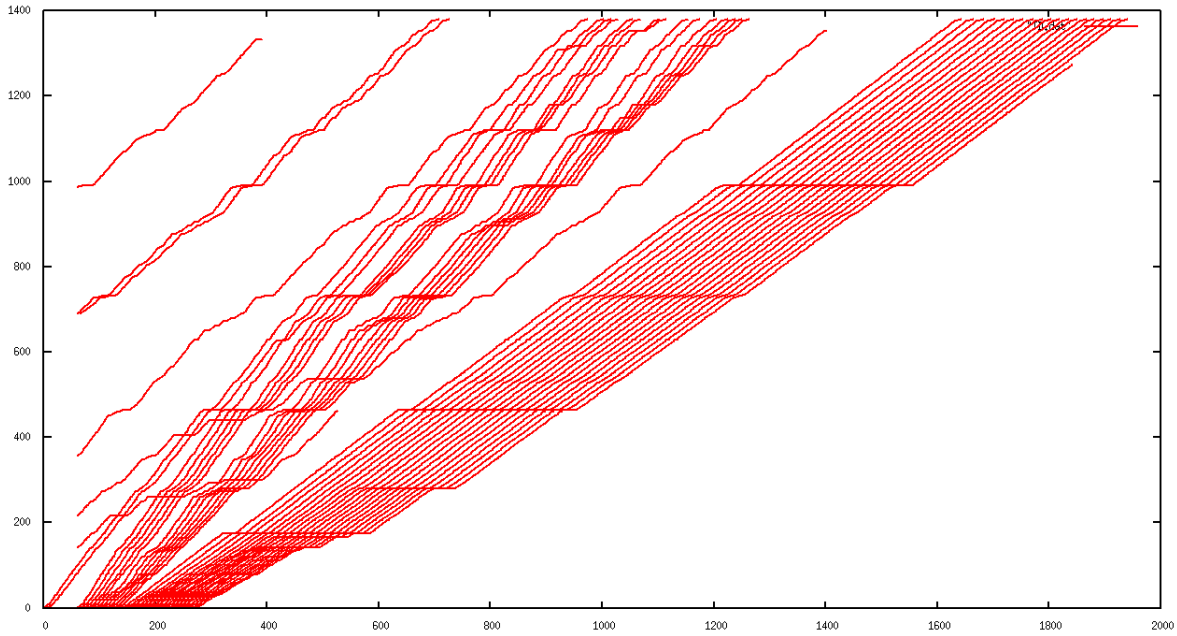

Figure 3.7: Initial proposed schedule

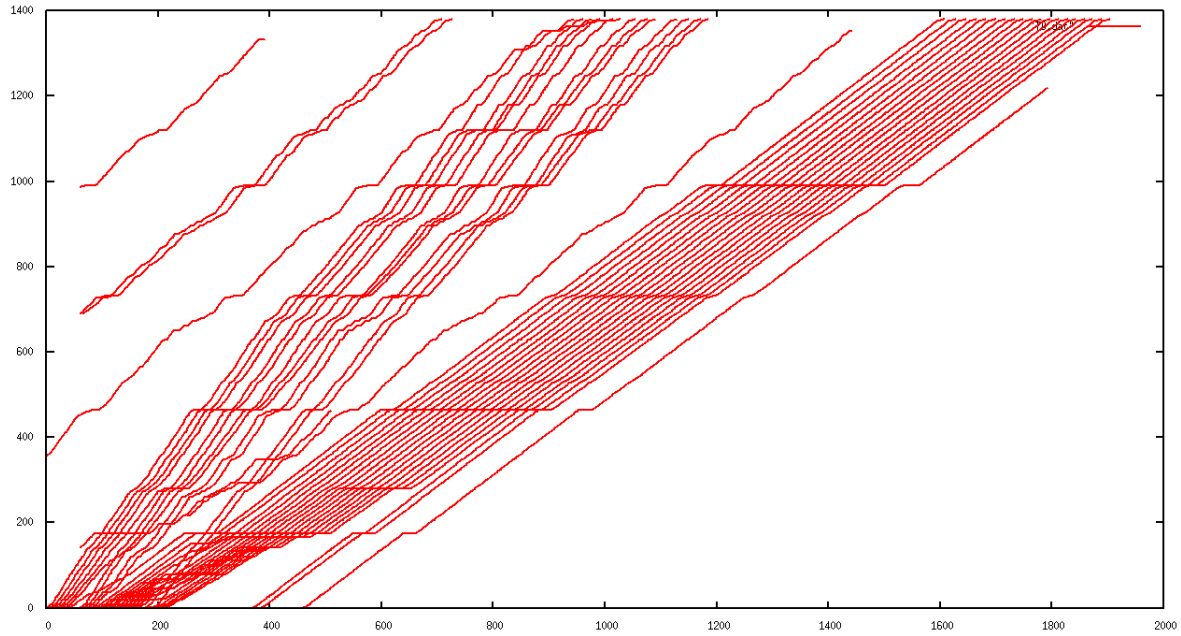

Figure 3.8: Distance vs Time graph after 30 iterations of SA

criterion has been satisfied. Hence the total number of iterations in the final graph are less than 100, as few of the epochs are dropped due to no improvement compared to previous epoch.

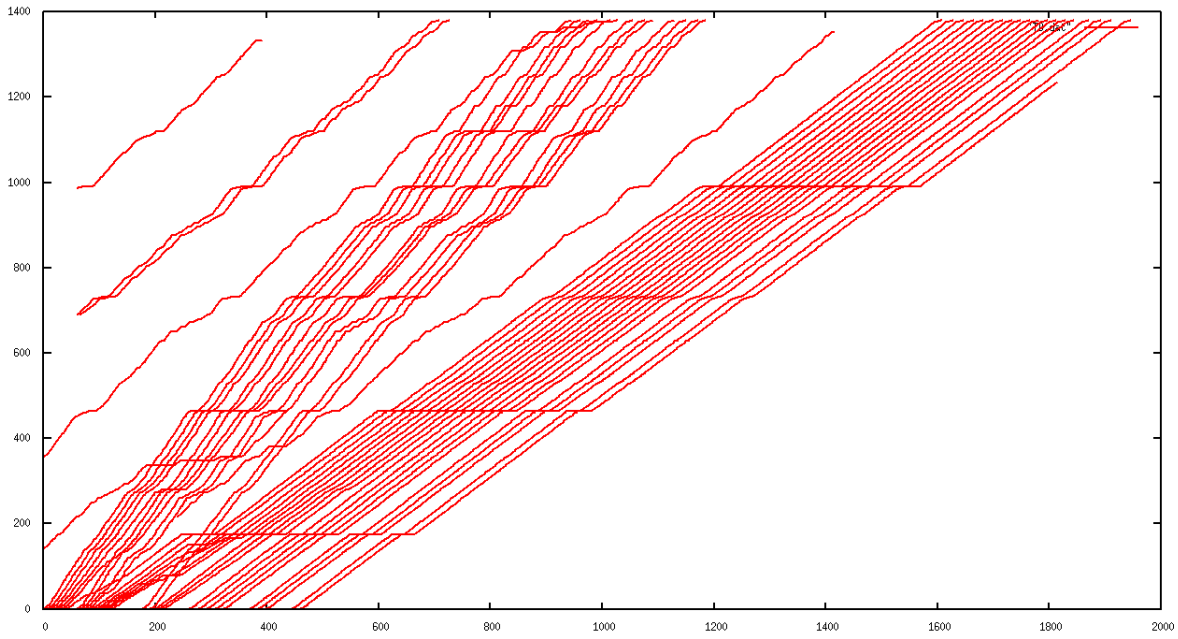

Figure 3.9: Simulation of train vs distance after 95 iterations of SA

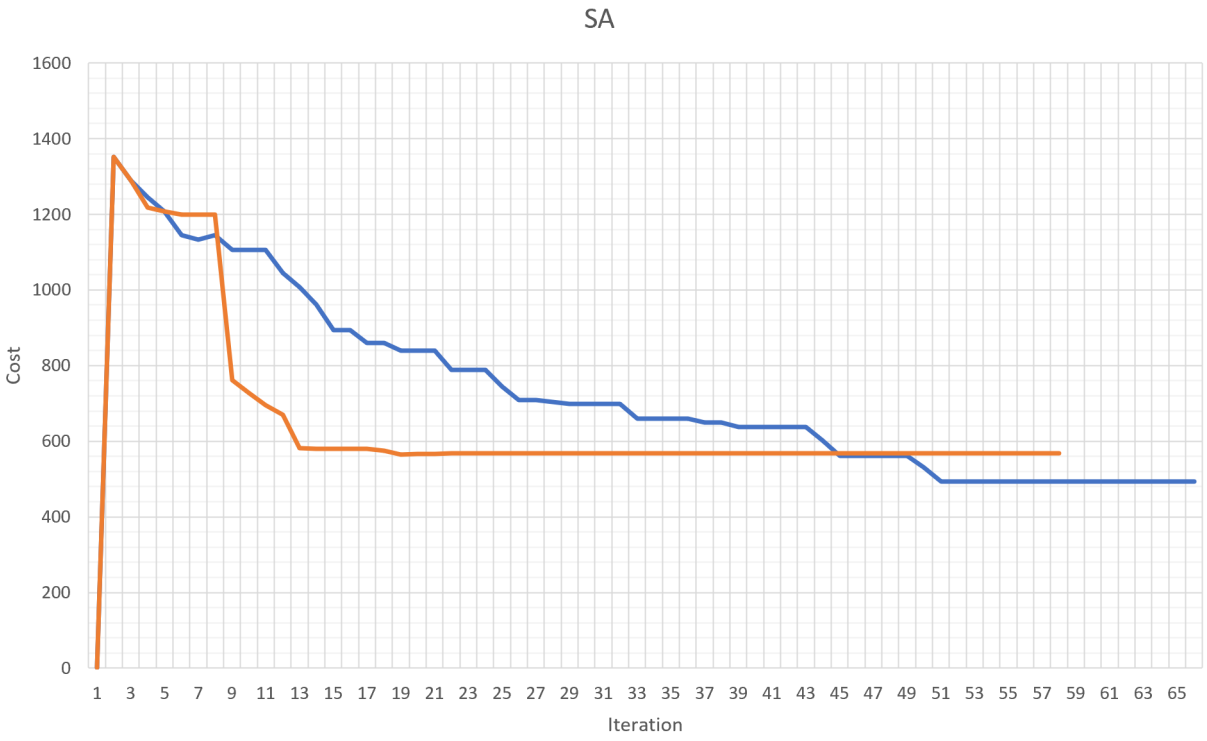

Figure 3.10: Comparison of 2 different SA simulations

/*TrainNo.	Initial	Final	difference	/*TrainNo.	Initial	Final	difference
11012001	60	60	0	11010001	60	0	-60
11012002	60	60	0	11010002	60	60	0
11012003	60	60	0	11010003	60	60	0
11012004	60	60	0	11010004	60	0	-60
11012005	60	60	0	11010005	60	18	-42
11012006	60	60	0	11010006	60	191	131
11012007	60	261	201	11010007	60	0	-60
11012008	60	278	218	11010008	60	177	117
11012009	60	292	232	11010009	60	0	-60
11012010	60	308	248	11010010	60	0	-60
11012011	60	323	263	11010011	60	0	-60
11012012	60	204	144	11010012	60	0	-60
11012013	60	0	-60	11010013	60	60	0
11012014	60	369	309	11010014	60	0	-60
11012015	60	384	324	11010015	60	60	0
11012016	60	399	339	11010016	60	174	114
11012017	60	214	154	11010017	60	60	0
11012018	60	199	139	11010018	60	60	0
11012019	60	444	384	11010019	60	60	0
11012020	60	460	400	11010020	60	0	-60

Figure 3.11: Initial and Final starting times of proposed schedule of 100 iterations of SA

Chapter 4

Summary and Conclusions

In this thesis,

- A framework was generated to modify the existing timetable through multiple stochastic paths to generate multiple timetables using an existing railway network simulator
- A metric that can be used for comparing two versions of timetables has been used to execute two learning algorithms (namely PIA and SA)

4.1 Scope for future research

- The current design of feedback does a linear allocation of importance to delays to all stations of a train uniformly. But in reality, the starting times of trains are dependent on the most occupied node (station). This concept can be associated with centrality in a graph. There is a degree of centrality associated with nodes of a graph, and delay at these nodes must be penalized more compared to the delay at other nodes
- The current framework can only modify the starting times of one train per iteration. A method of modifying the starting times of multiple of trains to be modified is to be designed.
- Once a large data-set of successful actions are generated, an actor-critic based neural network can be trained to estimate the non-linear relation between various features of trains such as priority, net delay, nearest delay etc. and the amount of delay/pre-pone time suggested for the trains.

- Currently for the freight window we have used multiple trains at low speed for generating the freight windows. But eventually we have to design a metric that rewards freight windows. This can complement the existing reward procedure of minimizing delay of coaching trains. This trade-off needs a careful characterization.

- The process has been paralyzed using make files. To further ease the procedure the csv files can be implemented using SQL database. This makes processing the huge data for exploratory analysis easier with less amount of code, thereby reducing errors.

References

[1] A. Punit, "Mechanised maintenance philosophy of railway assets." [Online]. Available: https://uic.org/IMG/pdf/d2_s4_mechanised_maintenance_philosophy_of_railway_assets_punit_agrawal_india_part2.pdf

[2] K. Sidhartha, I, "Network-wide mixed-rail traffic scheduler: challenges and implementation aspects," submitted for journal.

[3] P. I. Bureau, "Performance of indian railways in the last five years." [Online]. Available: https://pib.gov.in/PressReleaseIframePage.aspx?PRID=1602424

[4] H. Khadilkar, "A scalable reinforcement learning algorithm for scheduling railway lines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 727–736, 2019.

[5] N. Rangaraj and M. Belur, "A concept note for railway timetabling to rationalize and improve capacity utilization," no. 1-25, 2018. [Online]. Available: https://www.ee.iitb.ac.in/~belur/railways/niti/Rangaraj-Belur-NITI-Aayog-concept-noteJan2018.pdf

[6] A. Caprara, M. Fischetti, and P. Toth, "Modeling and solving the train timetabling problem," *Operations Research*, vol. 50, pp. 851–861, 10 2002.

[7] Y. Guo, "A reinforcement learning approach to train timetabling for inter-city high speed railway lines," in *2020 IEEE 5th International Conference on Intelligent Transportation Engineering (ICITE)*. IEEE, 2020, pp. 493–497.

[8] M. Belur, "RDSO overview." [Online]. Available: https://www.ee.iitb.ac.in/~belur/railways/RDSO/RDSO-code-documentation/Simulator/index.html

[9] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A gentle introduction to reinforcement learning and its application in different fields," *IEEE Access*, 2020.

[10] B. Douglas, “Reinforcement learning with matlab.” [Online]. Available: https://www.mathworks.com/content/dam/mathworks/ebook/gated/reinforcement-learning-ebook-all-chapters.pdf

[11] A. Alharin, T.-N. Doan, and M. Sartipi, “Reinforcement learning interpretation methods: A survey,” *IEEE Access*, vol. 8, pp. 171 058–171 077, 2020.

[12] S. Pavel, “Markov decision process.” [Online]. Available: https://www.coursera.org/learn/practical-rl