Reinforcement Learning for Train Movement Planning at Railway Stations

Adaptive and Learning Agents Workshop 2020

Shripad Salsingikar
Indian Institute of Technology Bombay
Powai, Mumbai, India
TCS Research, Tata Consultancy Services
Thane, India
shripad.salsingikar@tcs.com

Narayan Rangaraj
Indian Institute of Technology Bombay
Powai, Mumbai, India
narayan.rangaraj@iitb.ac.in

ABSTRACT

Train movement planning at a railway station involves assigning platforms to trains, finding traversal paths for trains entering or leaving through the station network, and scheduling train entry, exit, or reversal activities. Trains use railway tracks at the station for traversing through the station. The planning of train movements becomes challenging as a large number of decisions are involved, due to a large number of interconnected tracks at the station. Even for a small station, the problem is interesting due to some peculiar constraints and conditions. We develop a reinforcement learning-based approach to generate a train movement plan at the station. We consider route-lock route-release as operating policy, shunting/reversal activities, and route dependent run time while modeling the problem. The reinforcement learning algorithm learns best heuristics, which is used for planning train movements under different inter-arrival times between trains. In this paper, we demonstrate how this approach can be used to analyze the impact of inter-arrival time between trains on delay acquired by trains while traversing through a terminal station. We also demonstrate how this approach can be used to analyze the operating capacity of the terminal station and to learn heuristics for generating the train movement plan. We compare the results obtained by the heuristics learned by the reinforcement learning approach with the heuristics used in practice. We found that the heuristics learned by the reinforcement learning approach perform better in congested scenario compared to that being used in practice. Data instance used for the case study includes a small terminal station extracted from the Indian Railways network.

KEYWORDS

Reinforcement Learning; Train Planning; Junction Planning

1 INTRODUCTION

A station in a railway network is a place with multiple parallel tracks, which facilitate embarking and disembarking of passengers or loading and unloading of goods or both. A terminal is generally at the end of a railway line where the railway tracks end. It is a station at which trains can enter and leave in only one direction. Thus train reversal activities must be performed in railway terminals.

A railway station consists of interconnected track resources such as loops, interchanges, switches, and cross-overs along with platforms. A station handles train movements (arrival, halt, and departure), train reversal, and shunting movements as part of regular train operations. The shunting activities include movement of rolling stock (coaches, wagons) for sorting and creating new trains. Figure 1 gives a schematic layout of a typical railway station. A large railway station handles significant train movements that use track resources available at the station. These significant train movements and interconnected track resources cause hindrance in train movement, which leads to delays. Often, in a railway network, a large railway station is a cause of delay in the system.

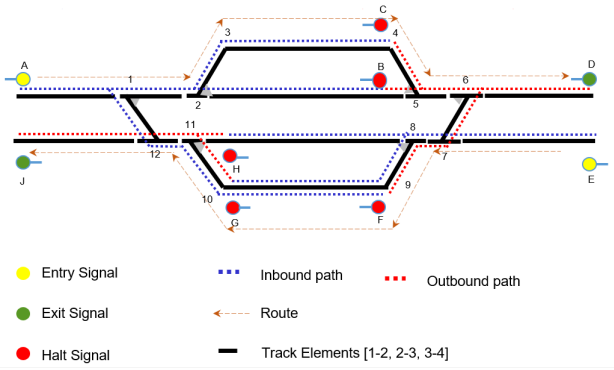


Figure 1: Station Schematic Layout

Train movement planning finds a conflict-free path for each train traversing through the station such that operational and safety requirements are satisfied, planned train movements are carried out smoothly, and overall delays are minimized. It includes scheduling trains, finding traversal paths for trains through the station network, assigning tracks and platforms to trains, and planning local reversal and shunting activities, if any. The outcome is represented as occupation start time and end time for each train for each allocated track element in the station. Key decisions involve determining a route for each train (a sequence of track elements visited by each train), determining track elements to be allocated to each train, time intervals for which track elements will be occupied by each train [14]. Planning of these activities at a railway station is challenging, especially in the situation of congestion, due to the number of decisions involved, details of infrastructure to be considered, and operational rules constraining the passage of trains through track sections [7].

Infrastructure details are defined in terms of track elements between signals/switches. The interconnection of track elements forms a railway network, and the sequence of track elements forms a path. A route is a path which a train traverses in the station. The origin of the path is the entry point of a train. The destination of the path is the exit point of a train. Please refer Figure 1 for details. Track elements at the station would be small in length, and a train may occupy more than one track element at a time.

Each train is defined in terms of an entry point and exit point at the station, scheduled arrival time at the entry point, scheduled departure time from the halt signal (i.e., the platform) at the station, and minimum halt time at the station. A set of trains and related activities (traversal, reversal) at the station are assumed to be known in advance.

For safety reasons, trains cannot move faster than the maximum permissible speed of the track elements, especially at crossings/switches. No infrastructure element is allowed to be allotted to two different trains at the same time. The traffic approaching towards already occupied tracks from other connected tracks is blocked. Minimum time between two trains must be maintained when they occupy the same resource consecutively.

How a train traverses through a station network depends on the underlying interlocking system and related constraints. These constraints decide how tracks are reserved ahead and released later during traversal of the train through the station. In *route-lock route-release* interlocking setting, all path elements for a train are reserved simultaneously, and all path elements other than the last element (platform or halt edge) are released simultaneously when the train reaches the last element. The other setting are *route-lock sectional-release* and *sectional-lock sectional-release* [20].

In this paper, we present a Reinforcement Learning (RL) based approach for developing heuristics, which are then used to create an operational train movement plan for a station. The output of the RL approach is operating heuristics similar to the heuristics used by the planners in real-life operations. We present a case study where we use the proposed RL algorithm to plan the train movements for a real-world instance extracted from the Indian Railways network. We analyze the impact of inter-arrival time between trains on train movements through the station. The inter-arrival time is the time between two consecutive trains entering the terminal. In this paper, it is the minimum time gap, after which the next train can enter the terminal. We report key performance parameters such as acquired delay, entry delay, a suitable range of inter-arrival time between trains for operations, and utilization of bottleneck resources. We compare the train movement plan developed using the proposed RL approach and the train movement plan developed using operational heuristics used in practice.

The outline of this paper is as follows. Section 2 presents a brief literature review. Section 3 describes the approach in detail. Section 4 presents a case study, where the proposed approach is applied to a realistic railway network, followed by conclusion in Section 5.

2 PREVIOUS WORK

The railway scheduling problem is a variant of job shop scheduling problem with blocking and no-wait constraints [9], where a track corresponds to a machine, and a train corresponds to a job. A train must wait at the current track unless the next track is freed, which makes tracks as machines with no-store / no buffer. The railway scheduling problem is the NP-Complete problem [1]; thus, various approaches are presented in the literature to solve this problem. The solution approaches for the railway scheduling problem can be classified into three types, namely (Meta) Heuristic, Analytical, and Simulation [8]. Recently machine learning approaches are being applied.

In the railway station planning problem, the underlying network is limited to the vicinity of a railway station. The characteristics of the railway station planning problem include dense train traffic, many smaller track elements interconnected with each, smaller planning horizon. The authors consider both tactical as well as operational settings while solving the problem. In tactical settings, planning is done well ahead of the implementation of the plan (timetable generation). In operational settings, planning is done close to the implementation of the plan (dispatching or operational rescheduling). In tactical settings, the sub-problems such as scheduling, routing, platforming, shunting are solved, either separately or in an integrated manner. Caprara et al. [3], [4] have studied routing and platforming problem separately, whereas Carey et.al. [5] and Dewilde et.al. [11] have studied the integrated scheduling, routing and platforming problem. Chakroborty [6] gives MILP model for platform allocation for an India railway instance. Most of the papers model railway infrastructure at the mesoscopic level. Lusby et al. [18] give a detailed review of approaches used for tactical problems. All the above authors have used exact optimization methods as they considered tactical settings.

In the operational settings, the research on real-time routing and platforming in railway junctions has gained importance recently with the emphasis on using exact optimization methods. The reported papers in the literature differ in the granularity at which the infrastructure is modeled, speed of trains considered (fixed or variable), routing (fixed, changing), planning horizon considered. Dessouky et al. [10], Törnquist et al. [25] and D'Ariano et al. [9] model the railway infrastructure at some higher level of granularity and use fixed routes and fixed speed for trains. Rodriguez et al. [21], Lusby et al. [19] model detailed level of infrastructure, in terms of track circuits and uses variable speed for trains. Caimi et al. [2] model the microscopic network details and use a fixed speed model. Fang et al. [13] give a review of rescheduling approaches.

Some of the recent papers in railway rescheduling at station report on analysis of various operating policies followed at the junction. Corman et al. [7] considers an interlocking system and compares route-lock route-release and section-lock section-release policies by keeping the routes of the trains unchanged. Pellegrini et al. [20] compare interlocking settings with changing the routes and found that route-lock section-release gives better results when trains travel along only partially coincident routes. In both the above studies, the speed of trains is assumed to be constant, and both use the exact optimization approach to solve the problem.

Recent papers apply the machine learning approach for railway scheduling. Hirashima [15] presents a reinforcement learning approach for planning train marshaling within yards. Dündar et al. [12] use supervised learning for mimicking human controllers but is limited to conflict resolution. Khadilkar et al. [17] present a supervised learning approach that learns from the solutions obtained

using a mixed-integer linear programming approach on small data instances. They derive a set of rules using decision trees and apply the scheduling rules to large test instances. Šemrov et al. [26] present reinforcement learning for railway scheduling. They use a Q-learning approach with the focus on recovery from initial delays. Khadilkar [16] presents an algorithm for scheduling bi-directional railway lines (both single- and multi-track) using a reinforcement learning approach used for timetable generation. This approach can scale to large, realistic, single- and multi-track instances of railway scheduling.

The problem studied in this paper is similar to that studied in Corman et al. [7] and Pellegrini el al [20], except that both authors study the problem in rescheduling settings, whereas we use timetabling settings. Both above authors [7] and Pellegrini el al [20] approximate the problem by fixing some of the variables and limits the planning horizon or consider rolling time horizon approaches. We consider a more generic setting by not fixing any variable or limiting the planning horizon.

This paper describes a reinforcement learning approach, which produces results better than heuristic approaches using relatively low computation times. The approach belongs to a class of reinforcement learning algorithms known as table-based Q-learning [7]. To our best knowledge, this is the first paper that uses reinforcement learning for solving train movement planning at a railway station. Both Šemrov et al. [26] and Khadilkar et al. [16] present an RL approach applicable to a long-distance railway line and not for the small area network like as a railway station. Both consider train occupancy at a block section as the state space, but here in the case of the station network, the block section is not clearly defined. At a station, a block section may not be the tracks between two signals. Thus, train occupancy by path need to be considered, and a path may have multiple signals or even no signal in two paths combined. This paper explicitly models a) route-lock route-release as interlocking setting, b) local shunting/reversal activities, and c) route dependent run time, which otherwise is not considered in the literature. This paper describes how reinforcement learning is used to learn heuristics automatically, which helps generate the schedule. This paper presents a computational experiment to prove the suitability of the proposed approach for determining the carrying capacity of the station, and to generate a timetable for trains in the station.

3 METHODOLOGY

This section describes details of the proposed reinforcement learning approach to generate the train movement plan at a station. We model the station network as a network of connected track resources; take expected arrival time of each train visiting the station in the given planning horizon along with its entry signal point and exit signal point in the network as input; and plan train movements through the station by considering train routing along all possible routes in the station; Moreover, find a conflict-free path for each train through the station by using the proposed RL approach.

The output is the operating schedule of the trains at the station, i.e., the best routing option and a feasible schedule for all trains under consideration, along with other key performance parameters such as entry delay, acquired delay, and resource utilization.

We first describe how infrastructure, train movements are modeled, and then the details of the reinforcement learning model developed.

3.1 Station Network and Train Movement Modeling

A railway station network consists of small interconnected elements like signals, crossings, and track segments. Figure 1 gives a schematic layout of a typical railway station. It explains concepts like track element, node/signal, inbound path, outbound path, route. A route is a traversal path for a train in the station. A route has an entry point as the origin and an exit point as destination. Please refer Figure 1 for details. An Inbound path is between an entry point signal and a halt signal, whereas an outbound path is between a halt signal and an exit point signal.

For this study, we model the track infrastructure of the station at the microscopic level. We represent station network as a directed graph with edges representing track elements and nodes representing signals or start/end of track elements. A node is characterized by a name, node type (signal/crossing), and the direction of traffic allowed through that node. The signal node is classified as entry, exit, halt, intermediate, and station signal. An arc is characterized by from-node, to-node, length of the arc/track, speed limit while traversing through the arc, arc type (regular or crossing), and direction of traffic allowed through the arc. An arc is a directed arc representing the allowed direction for the train traffic.

We obtain all possible routes between all pairs of entry signals and exit signals (in the directed graph shown in Figure 1), using the 'depth-first graph traversal' algorithm. There may be multiple routes possible between a given entry-signal and a given exit-signal. We discard all routes with user-defined conflicting links between entry node and exit node and route with no halt signals. Further, each route is broken into two paths, the inbound path, and the outbound path. An inbound path is between an entry signal and a halt signal, whereas an outbound path is between a halt signal and an exit signal.

Each train is defined by train name, entry point, exit point, expected arrival time at the entry point, entry speed, length, acceleration, deceleration, minimum halt time, and if it requires a reversal or not.

We assume that trains are running at a constant speed, but we model different speeds for different paths and routes of trains. We calculate all the possible routes which a train may take in the station. Further, we assume route-lock route-release as the interlocking setting, and we model the train traversal accordingly.

3.2 Reinforcement Learning model

State-Space representation The railway track network at the station is represented using paths and platform tracks, as shown in Figure 1. At a station, let 'NS' be the number of entry signals (i.e., locations from which trains can enter the station or station), 'PI' be the number of inbound paths (i.e., paths from entry signals to platforms or halt signals), 'PO' be the number of outbound paths (i.e., paths from platforms or halt signals to exit signals) and 'HS' be the number of halt signals, which also represents the number of platforms at the station.

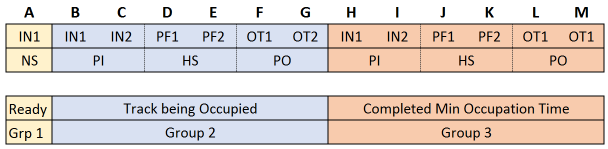


Figure 2: State space representation

A state vector contains three types of entries, and all entries are binary; see Figure 2 for details. The first group of entries indicates whether a train is ready to enter the station at an entry signal. The length of this part is equal to NS. The value 1 indicates that a train is ready to enter the station, 0 if not.

The second group of entries represents if tracks (paths or platforms) are occupied or not. The length of this part is equal to the sum of PI, HS, and PO. This group has three subparts. First, PI entries represent occupancy of inbound paths, the next HS entries represent occupancy of platforms and the next PO entries represent occupancy of outbound paths. 1 represents that a path is occupied, 0 otherwise.

The third group of entries represents if train occupying respective tracks completed its minimum occupation time (defined by minimum run-time or minimum halt-time of a train) Moreover, that tracks are ready to be released. The length of this part is equal to the sum of PI, HS, and PO. This group has three subparts where the first PI entries represent release readiness of inbound paths, the next HS entries represent release readiness of platform, and the next PO entries represent release readiness of outbound paths. One represents that a path is ready for release, Zero otherwise. Please note that a path or a platform can still be occupied by a train, even its minimum occupation time is over, and it represents delay.

Action and Policy representation A train in the station traverses through a set of tack resources in a specified order. A train enters the station via an entry signal, accesses inbound path, halts at a platform, and exists the station using an outbound path via an exit signal. In this study, we represent which track resources (path or platform) will be occupied next by the train as an action for a given state vector. The action represents traversal of trains from one resource to another. The transition from an inbound path to a platform is an action. The transition from a platform to an outbound path is an action. For example, if a train is ready to enter the station and no other train is present in the station, then the train has to choose one of the inbound paths, this choice is an action in this study. The action space depends on the underlying station network and consists of a set of inbound paths, platforms, outbound paths, and exit signals. For simplicity and clarity, we define an action as a combination of the previous and the next resource.

For many states, only one action is possible, whereas, for a few states, multiple actions are possible. The RL procedure maps each state vector to a probability of choosing the action to be taken. The final goal is to identify one best action for each possible state vector for a given infrastructure. This set of state-action pairs, which is expected to gives the best result, is referred as policy.

Objectives, Rewards, and Q-Values For this study, we use the objectives of minimizing the make-span (i.e., the difference between the departure of the last train and the arrival of the first train). To make the learning fast, we do not directly use the make-span as the (negative) reward at the end of the episode, However, we use a reward system given in [16]. The algorithm maintains a threshold of J as the goal to be achieved in each episode. J is the average of objective values of last ρ episodes, where $\rho > 0$ and is a user-defined parameter. The threshold becomes tighter as the best known J is improved upon during learning. A reward of +1 (success) is given if the make-span at the end of the episode is under the current threshold, and -1 (failure) if it is over the threshold.

Reinforcement Learning Algorithm We get a set of unique state-action pairs from a state vector and its associated actions. Each state-action pair (s, a) is associated with a Q-Value $q(s, a)$. The higher the Q-Value, the higher the desirability of the relevant pair.

For this study, we use ϵ-soft On-Policy First Visit Monte Carlo Control algorithm, which is adapted from [24]. The On-Policy Monte Carlo algorithm aims to shift the policy toward a deterministic optimal policy. In ϵ-soft policy most of the time, we choose an action that has maximal estimated action Q-value. With probability ϵ, we instead select an action at random from the available/possible actions. We start with a random policy where we choose one action randomly for each state-action pair. The policy chooses the greedy option (higher Q-Value) with probability (1-ϵ), and a randomized action with probability ϵ. The value of ϵ starts at 1 in the first training episode and decreases as more episodes are completed. This gradual decrease in ϵ moves the policy gradually from exploration towards exploitation.

In each episode, we track only pairs belonging to states where multiple actions are possible and not the entire sequence of state-action pairs. The rewards received at the end of the episode are discounted, using a discounting factor γ as per the first-visit policy given in [24]. We use table-based Q-Learning, i.e., Q-value for each state-action is store in a table.

4 CASE STUDY

This section presents a case study and experiments performed on a real-life small terminal station extracted from the Indian Railways network. The traffic and complexity of this instance are nearly the same as happens in real life. We have used a small terminal station for this case study. There exist medium to very complex station network layouts in practice as given in [22], [23], [7], and [20]. The proposed approach with some modification is also applicable to these bigger station networks.

We use the RL approach described in section 3 to develop heuristics, which are used for developing an operational train movement plan for a station. These heuristics are similar to the ones used by planners/users in real-life operations.

The experiment demonstrates how the proposed RL approach can be used for determining a timetable for the trains running in the system. The experiment also demonstrates the impact of various headway values (time between two consecutive trains entering the terminal) on the overall delay, which in turn determines the capacity of the terminal station. Delay includes delay acquired by trains in the system as well as delay experienced by trains at the entry of the system.

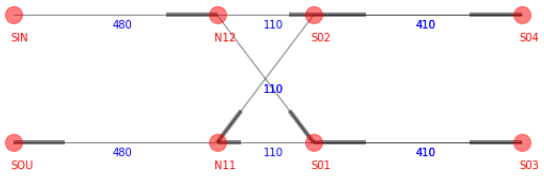


Figure 3: CSMT Harbour terminal scissor layout

Table 1: Runtime data for scissor layout

Route-Path	Edges	Runtime (sec)	Total (sec)
R1-P1	SIN - N12 - S01 - S03	40 - 25 - 095	160
R1-P2	S03 - S01 - N11 - SOU	25 - 10 - 045	80
R2-P3	SIN - N12 - S02 - S04	35 - 10 - 035	80
R2-P4	S04 - S02 - N11 - SOU	55 - 25 - 100	180

4.1 Terminal Network Layout

The Chatrapati Shivaji Maharaj Terminal (CSMT) - Harbour is used as an illustrative example. It is the starting/terminal station of Harbour commuting local line in Mumbai, operated by Central Railway of Indian Railways and is a conventional double line.

CSMT Harbour terminal has a simple network layout with some peculiar settings, which makes it interesting for study. Due to the network layout of the terminal, the travel time of a train depends on which route train takes in the terminal, and thus, the minimum time between two consecutive trains is not always the same. Although the minimum time between every alternate train entering the terminal is the same, the minimum time between two consecutive trains entering one after each other in the terminal does not remain the same.

Figure 3 gives layout of CSMT Harbour terminal. The length of the arc (in meter) is shown just above the arc. The thick end of an arc represents the direction of traffic allowed on that arc. Nodes starting with 'S' are signals, whereas that with N' are crossings.

The CSMT Harbour terminal has only two terminating lines. These two lines are interconnected with each other at the start of the terminal for facilitating train reversal movement. A train enters the terminal using the UP line (SIN signal) and leaves the terminal using the DOWN line (SOU signal). The crossing is involved when trains enter or exit the station, depending on the line selected for trains for halting and reversing.

There exist four paths and two routes in this terminal. Table 1 gives the list of possible paths and corresponding run-time for each path in the terminal. The traversal time is pre-calculated using standard motion equations taking entry speed, speed-limit at tracks, acceleration, deceleration of train. The arc N12-S01 connects Entry-Signal SIN and Platform S01-S03 in up direction and is in path P1. The arc S02-N11 connects Platform S02-S04 and Exit-Signal SOU in the down direction and is in path P4. These two arcs cross each other physically. Route R1 consists of path P1, path P2, platform S01-S03, and arc N12-S01. Route R2 consists of path P3, path P4, platform S02-S04, and arc S02-N11.

Please note that traversal time for route R1 is less than that of route R2. This difference is because tracks after the crossing have the right curvature.

4.2 Experiment setup

State space: For the network layout given in Figure 3, the state space vector is defined as given in section 3.2. The layout consists of two inbound paths, two outbound paths, two platforms, and one entry signal. The length of the state-space vector is 13. The length of the first group representing entry points is 1. The length of the second group representing train occupancy of paths/tracks is 6. The length of the third group representing readiness of train leaving paths/tracks is 6. Figure 2 illustrates the state space representation for this case study.

The state-space vector can be represented as a string "A-BCDEFG-HIJKLM," where each element is binary. The value of A is 1 if any train is ready to enter the terminal from signal SIN, else 0. B and C represent if inbound-path1 and inbound-path2 are occupied, respectively. If a path is occupied, the respective variable (B and C) gets value 1, else 0. D and E represent if platform1 and platform2 are occupied, respectively. If a platform is occupied, the respective variable (D and E) gets value 1, else 0. F and G represent if outbound-path1 and outbound-path2 are occupied, respectively. If a path is occupied, the respective variable (F and G) gets value 1, else 0.

H and I represent if inbound-path1 and inbound-path2 are ready to release, respectively. When the train occupying respective inbound path, completes minimum time for occupation (defined as per train's run time on that path), respective variable (H and I) gets value 1, else 0. J and K represent if trains occupying platform1 and platform2 are ready to release the respective platform. When the train occupying the respective platform, completes minimum time for occupation (defined as per train's halt time at the terminal), respective variable (J and K) gets value 1, else 0. L and M represent if outbound-path1 and outbound-path2 are ready to release, respectively. When the train occupying respective outbound path, completes minimum time for occupation (as per train's run time on that path), respective variable (L and M) gets value 1, else 0.

Action Space: Some example of actions are given here. NONIN1 - train entering from entry signal SIN to inbound path1; IN1PF1 - train on inbound path1 will occupy platform1; PF1OT1 - train on platform1 will occupy outbound path1; OT1NON - train on outbound path1 will exit the terminal. Similarly, actions NONIN2, IN2PF2, PF2OT2, OT1NON are defined. NONNON action is for no actions or invalid moves.

Converting a RL policy to heuristics In the approach presented in the paper, the state-space represents the current occupation of track resources and the ready-ness of trains entering the junction or leaving the currently occupied track resources. The action-space represents which track resources which would be occupied next by the train for a given state. After running the experiment, we get the optimal RL policy. In the RL policy, for each state in the state-space vector, we get an action that will result in the best solution. In turn, for a given resource occupancy (state), we would get the following resources to be occupied as a heuristics.

Test Instances: Each data instance consists of 25 homogeneous trains where each train is having a network entry time determined

by the headway parameter. We assume that all trains are available at a deterministic constant headway to enter the terminal. The time between two consecutive trains at station entry is constant. We generate 21 data instances, independent of each other, by varying headway in the step of 30 seconds starting from 0 seconds to 600 seconds. In the 0 seconds headway instance, all trains are available to enter the train at time 0. In the 180 seconds headway instance, a train is available to enter the station every 180 seconds.

4.3 Experiments

The experiments include generation of train movement plan for a terminal station with the layout given in Figure 3, using the run-time data given in Table 1 and the RL model described in section 3.2.

Several interesting questions are posed for analysis, as well as determining the operating heuristics for a terminal layout. The experiment aims to find the answers to these questions. These questions are related to platform allocation, and movement sequencing are

- If both platforms are free, then which platform should be allocated first?
- If both platforms are occupied, then which platform should be vacated first?
- If one platform is free and another occupied, then what should be done?
  - Should the entering train be allocated to the free platform?
  - Should the entering train wait for the occupied platform to be freed?

For comparison purposes, we define an operating heuristic used in practice. These heuristics are based on the experience and railway domain know-how of the railway planner and are used for planning train movement at the CSMT Harbour terminal in day to day practice. We formalize/derive these heuristics after our discussions with railway planners.

Operating Heuristics used in practice:

- If both platforms are free and both inbound paths are free, and a train is ready to enter, then Platform1 should be chosen first for allocation.
- If both platforms are occupied
  - If both trains are ready to depart then the train at Platform2 should be departed before any train is allowed to enter the terminal
  - Else (i.e., only one train is ready to depart) the train ready to depart should depart immediately
- If one platform is occupied and train at that platform is ready to depart, then the train ready to depart the terminal should be allowed to depart before the next incoming train is allowed to enter the terminal.

We conduct two sets of experiments. The first set of experiments is to determine the optimal RL policy and determining operating heuristics for each data instance. The second set of experiments is to determine a single global RL policy and determining a single set of learned operating heuristics applicable for all 21 data instances. For determining the optimal RL policy for each data instance, we run the proposed RL algorithm on each data instance for 5000 episodes Moreover, find an optimal policy for each data instance. For determining a single global optimal RL policy, we run the proposed RL algorithm on all 21 data instances for 125000 episodes. For each episode, a data instance is chosen randomly from the available 21 instances. For both these experiments, we use discounting factor $\gamma = 0.99$ Moreover, gradually decrease ϵ from 1 to 0 with the factor of $1/TotalEpisodes$ after each episode, where ϵ is the probability of choosing random action for a state.

The RL model is implemented in python3.6. All experiments are conducted on a computer with Intel(R) Core(TM) i5-2520M processor with 2.5GHz clock speed and 4.0 GB RAM running on Ubuntu 16.04 LTS.

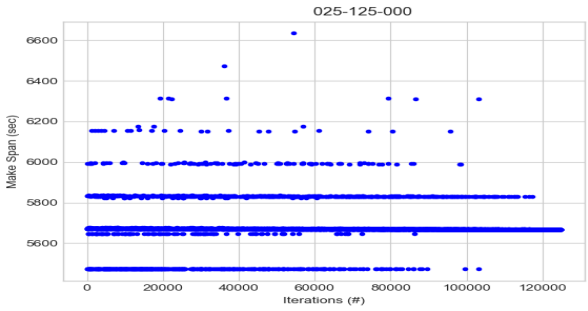
Table 2: Output objective (Make-span) values

Head-way (sec)	Practical Global (sec)	Converge Global (sec)	Converge Local (sec)	Observe Best (sec)
000	7,458	5,667	5,472	5,472
030	7,458	5,667	5,472	5,472
060	7,458	5,667	5,472	5,472
090	7,458	5,667	5,472	5,472
120	7,458	5,667	5,472	5,472
150	7,458	5,667	5,472	5,472
180	7,458	5,667	5,477	5,477
210	5,510	5,667	5,507	5,507
240	6,180	6,195	6,180	6,180
270	7,693	6,900	6,947	6,900
300	7,756	7,635	7,756	7,620
330	8,446	8,355	8,446	8,340
360	9,136	9,075	9,136	9,060
390	9,826	9,780	9,826	9,780
420	10,515	10,691	10,515	10,515
450	11,220	11,235	11,220	11,220
480	11,940	11,955	11,940	11,940
510	12,660	12,675	12,660	12,660
540	13,380	13,395	13,380	13,380
570	14,100	14,115	14,100	14,100
600	14,820	14,835	14,820	14,820
Total	195,388	182,177	180,741	18031

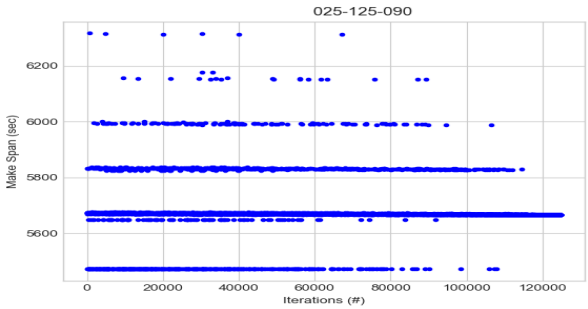
4.4 Results:

The 'Practical Global' column of Table 2 shows the objective value obtained for each data instance using the operating heuristics used in practice (given in the previous section). The objective values for 'Practical Global' for headway less than 180 seconds are the same. Surprisingly this value is higher than the value for the headway of 210 seconds. It may be because practitioners never experienced the headway less than 210 seconds in practice.

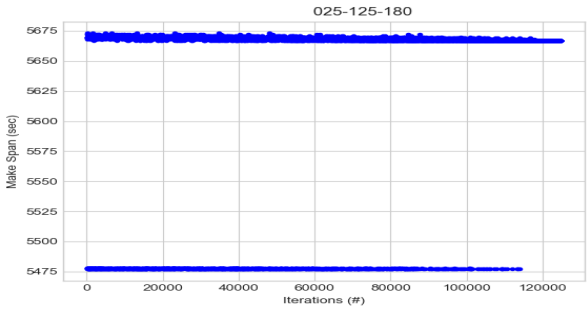
Optimal RL policy for each data instant: The 'Converged Local' column of Table 2 shows the objective value obtained for each data instance for the converged RL policy. The 'Best-Observed Local' column of Table 2 shows the best objective value observed for each data instance during the experiment. For headway between 270 seconds to 390 seconds, the RL algorithm does not converge to the best value of the objective. The objective value for the 'Conversed Local' policy is better than or equal to that of the 'Practical Global' policy for all data instances. Although we derive the operating



(a) Headway = 0 min

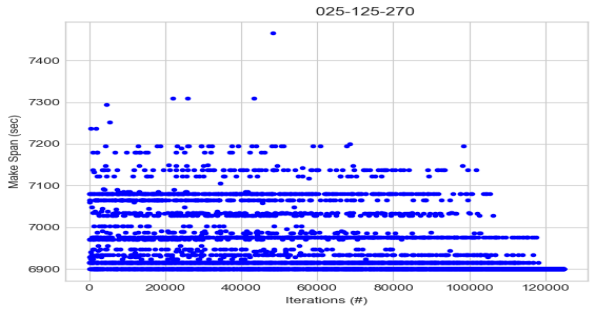


(b) Headway = 90 min

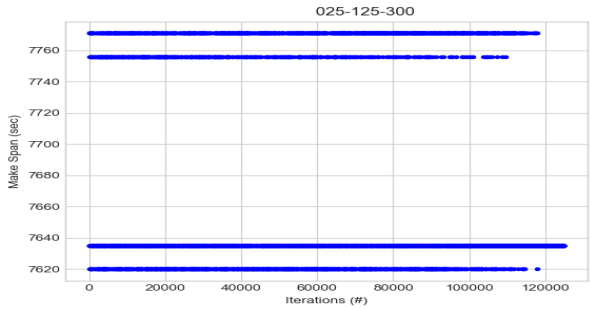


(c) Headway = 180 min

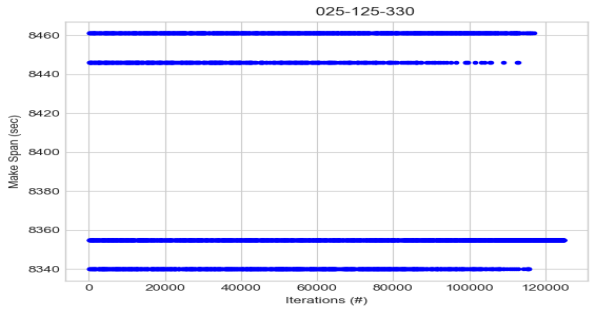
Figure 4: Learning Curve - Low Headway



(a) Headway = 270 min



(b) Headway = 300 min



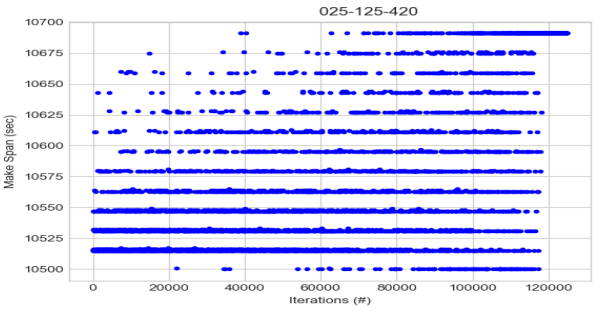
(c) Headway = 330 min

Figure 5: Learning Curve - Medium Headway

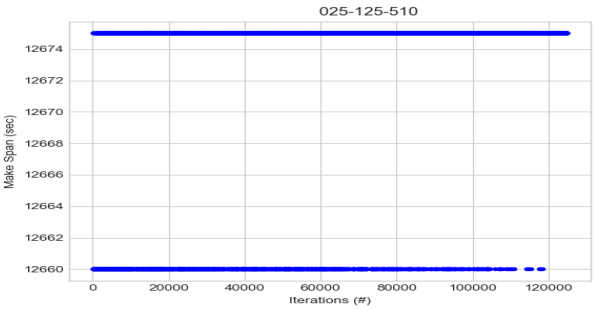
heuristics from the optimal RL policy for each data instance, they are not reported here.

Global RL policy for all instances: The 'Converged Global' column of Table 2 shows the objective value obtained for the converged global RL policy. The objective value for the converged global RL policy is better than or equal to the objective values obtained from the operating heuristics used in practice (given in column 'Practical Global') for all data instances. For headway between 270-390 seconds, it is even better than the converged RL policy for individual data instance. Figure 4, Figure 5, and Figure 6, give the learning curve obtained for respective headway values while running RL algorithm for obtaining global RL policy for all data instances. Each sub-figure depicts the objective value for the given headway at the iteration at which that headway was selected randomly. The title of each sub-figure represents the number of trains, the number of iterations in thousands, and the headway in seconds. The last 5,000 iterations in each sub-figure show the converged value. The parallel lines in the sub-figures indicate that the limited number of solutions for the problem at the respective headway.

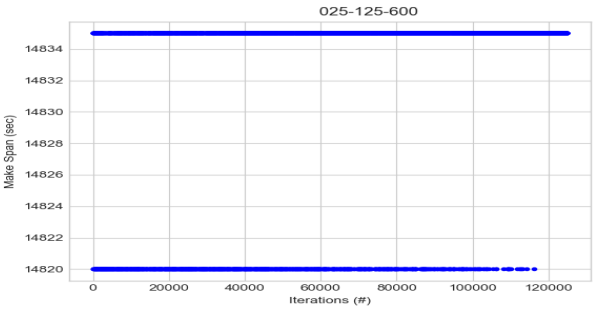
We derive global operating heuristics from the global RL policy obtained. Using these learned heuristics, we derive a timetable for trains running in the terminal for any given headway. The learned heuristics give better results than those used in practice, especially



(a) Headway = 420 min



(b) Headway = 510 min



(c) Headway = 600 min

Figure 6: Learning Curve - High Headway

when the headway is low, i.e., when congestion is high. The learned heuristics are given below, and these heuristics differ from those used in practice (given in the previous section).

- If both platforms are free and both inbound paths are free, and a train is ready to enter, then Platform1 should be chosen first for allocation. (Same as practice)
- If both platforms are occupied
  - If both trains are ready to depart, then train at Platform1 should be departed before any train is allowed to enter the terminal. (Differs from practice)
  - Else (i.e., only one train is ready to depart) the train ready to depart should be departed immediately. (Same as practice)
- If one platform is occupied and train at that platform is ready to depart and a train ready to enter the terminal, then the train ready to enter the terminal should be allowed to enter before any outgoing train is allowed to depart from the terminal. (Differs from practice)

Objective values for global RL policy are more than that of the optimal RL policy for each data instant, except for data instances with the headway of 270 to 390 seconds, where optimal RL policy for each data instant did not converge to best observed objective value.

The total of the objective values for the global RL policy is 1.02% move compared to the total of the objective values for optimal RL policy for each instance. From the operational point of view, it is better to have a single set of operating heuristics than having a separate operating heuristics for each data instance.

The above approach can also be used to determine the capacity of the terminal station. Observe that the objective values for both global RL policy and optimal RL policy for each instance are the same for headway between 0 seconds to 150 seconds, and it starts increasing from 180 seconds. It means that the CSMT terminal station can not handle the headway less than 180 seconds without delaying the trains at the entry. The CSMT terminal can not handle more than 20 trains per hour. In other words, the max capacity of the CSMT terminal station is 20 trains per hour.

5 CONCLUSIONS

We developed a reinforcement learning-based approach for generating a timetable for trains traversing through a railway terminal. We demonstrated how the reinforcement learning approach is used to learn heuristics, which in turn are used to generate a train movement plan. We compared the quality of the plan generated using rules used in practice and the plan generated by the heuristics learned using the reinforcement learning approach proposed in this paper, under different congestion scenarios. Based on the analysis, we conclude that different operating rules/heuristics are required to be used under different inter-arrival times (headway), i.e., under different congestion levels. In lower congestion (higher headway) situations, both practical rules and RL learned rules performed the same, but in higher congestion level, RL learned rules performed better. We also showed that a single global policy could be learned and used under multiple congestion scenarios. The quality of the plan generated by the single global policy is inferior compared to the quality of the plan generated by the different optimal policy for each data instance by less than 1%.

Future works include testing the Deep-Q-Learning based reinforcement learning approach at bigger station networks given in Salsingikar et al. [22] and Shekhar et al. [23]. Future work would include exploring route-lock section-release and section-lock section-release strategy.

ACKNOWLEDGMENTS

We thank Mr. Hardik Meisheri, Dr. Harshad Khadilkar, and Dr. Siddhartha SenGupta from TCS Research & Innovation unit of Tata Consultancy Services for their help and guidance during this work.

REFERENCES

[1] X. Cai and C. J. Goh. 1994. A fast heuristic for the train scheduling problem. *Computers and Operations Research* 21 (1994), 499–510.

[2] Gabrio Caimi, Martin Fuchsberger, Marco Laumanns, and Marco Lüthi. 2012. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research* 39, 11 (2012), 2578–2593.

[3] Alberto Caprara, Laura Galli, Leo Kroon, Gábor Maróti, and Paolo Toth. 2010. Robust Train Routing and Online Re-scheduling. In *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10) (OpenAccess Series in Informatics (OASIcs))*, Thomas Erlebach and Marco Lübbecke (Eds.), Vol. 14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 24–33. https://doi.org/10.4230/OASIcs.ATMOS.2010.24

[4] Alberto Caprara, Laura Galli, and Paolo Toth. 2011. Solution of the train platforming problem. *Transportation Science* 45, 2 (2011), 246–257.

[5] Malachy Carey and Sinead Carville. 2003. Scheduling and platforming trains at busy complex stations. *Transportation Research Part A: Policy and Practice* 37, 3 (2003), 195–224.

[6] Partha Chakroborty and Durgesh Vikram. 2008. Optimum assignment of trains to platforms under partial schedule compliance. *Transportation Research Part B: Methodological* 42, 2 (2008), 169–184.

[7] Francesco Corman, Rob MP Goverde, and Andrea D'Ariano. 2009. Rescheduling dense train traffic over complex station interlocking areas. In *Robust and Online Large-Scale Optimization*, Ravindra Kumar Ahuja, Múohring Rolf H., and Zaroliagis Christos D. (Eds.). Vol. 5868. Springer, Berlin, Germany, 369–386.

[8] Yong Cui. 2010. *Simulation based Hybrid Model for a Partially Automatic Dispatching of Railway Operation.* PhD Thesis. University of Stuttgart.

[9] Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. 2007. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research* 183, 2 (2007), 643–657.

[10] Maged M. Dessouky, Quan Lu, Jiamin Zhao, and Robert C. Leachman. 2006. An exact solution procedure to determine the optimal dispatching times for complex rail networks. *IIE transactions* 38, 2 (2006), 141–152.

[11] Thijs Dewilde, Peter Sels, Dirk Cattrysse, and Pieter Vansteenwegen. 2013. Robust railway station planning: An interaction between routing, timetabling and platforming. *Journal of Rail Transport Planning and Management* 3 (2013), 68–77.

[12] Selim Dündar and İsmail Şahin. 2013. Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. *Transportation Research Part C: Emerging Technology* 27 (2013), 1–15. https://doi.org/10.1016/j.trc.2012.11.001

[13] Wei Fang, Shengxiang Yang, and Xin Yao. 2015. A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems* 16, 6 (2015), 2997–3016.

[14] Ingo A Hansen and Jörn Pachl. 2014. *Railway Timetabling and Operations: Analysis, Modelling, Optimisation, Simulation, Performance Evaluation.* Eurorpress, Hamburg, Germany.

[15] Yoichi Hirashima. 2011. A reinforcement learning method for train marshaling based on movements of locomotive. *International Journal of Computer Science* 38 (2011), 242–248.

[16] Harshad Khadilkar. 2019. A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Transactions on Intelligent Transportation Systems* 20, 2 (2019), 727–736.

[17] Harshad Khadilkar, Shripad Salsingikar, and Sudhir Kumar Sinha. 2017. A machine learning approach for scheduling railway networks. In *7th International Conference on Railway Operations Modelling and Analysis*. International Association of Railway Operations Research, IFSTTAR, Lille, France, 302 – 316.

[18] Richard M Lusby, Jesper Larsen, Matthias Ehrgott, and David Ryan. 2011. Railway track allocation: models and methods. *OR Spectrum* 33, 4 (2011), 843–883.

[19] Richard M Lusby, Jesper Larsen, Matthias Ehrgott, and David M Ryan. 2013. A set packing inspired method for real-time junction train routing. *Computers & Operations Research* 40, 3 (2013), 713–724.

[20] Paola Pellegrini, Grégory Marlière, and Joaquin Rodriguez. 2014. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological* 59 (2014), 58–80.

[21] Joaquín Rodriguez. 2007. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological* 41, 2 (2007), 231–245.

[22] Shripad Salsingikar, L. Sathishkumar, Narayan Rangaraj, and Aseem Awad. 2017. Analysis and planning of train movements at a railway junction. In *7th International Conference on Railway Operations Modelling and Analysis*. International Association of Railway Operations Research, IFSTTAR, Lille, France, 541 – 556.

[23] Sudhanshu Shekhar, Abhishek Singh, Madhu N. Belur, and Narayan Rangaraj. 2019. Development of a railway junction simulator for evaluation of control strategies and capacity utilization optimization. In *2019 Fifth Indian Control Conference (ICC)*. 260–265. https://doi.org/10.1109/INDIANCC.2019.8715629

[24] Richard Sutton and Andrew Barto. 2018. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, United States.

[25] Johanna Törnquist. 2006. *Railway Traffic Disturbance Management.* PhD Thesis. Blekinge Institute of Technology.

[26] Darja Šemrov, Rok Marsetič, Marijan Žura, Ljupčo Todorovski, and Aleksander Srdic. 2016. Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological* 86 (2016), 250–267. https://doi.org/10.1016/j.trb.2016.01.004