Adaptive Railway Traffic Control using Approximate Dynamic Programming

Taha Ghasempournejad Seifdokht

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Centre for Transport Studies

Faculty of Engineering Sciences

University College London

December 8, 2019

I, Taha Ghasempournejad Seifdokht, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Railway networks around the world have become challenging to operate in recent decades, with a mixture of track layouts running several different classes of trains with varying operational speeds. This complexity has come about as a result of the sustained increase in passenger numbers where in many countries railways are now more popular than ever before as means of commuting to cities. To address operational challenges, governments and railway undertakings are encouraging development of intelligent and digital transport systems to regulate and optimise train operations in real-time to increase capacity and customer satisfaction by improved usage of existing railway infrastructure.

Accordingly, **this thesis presents an adaptive railway traffic control system for real-time operations based on a data-based approximate dynamic programming (ADP) approach with integrated reinforcement learning (RL)**. By assessing requirements and opportunities, the controller aims to reduce delays resulting from trains that entered a control area behind schedule by re-scheduling control plans in real-time at critical locations in a timely manner.

The present data-based approach depends on an approximation to the value function of dynamic programming after optimisation from a specified state, which is estimated dynamically from operational experience using RL techniques. By using this approximation, ADP avoids extensive explicit evaluation of performance and so reduces the computational burden substantially. In this thesis, formulations of the approximation function and variants of the RL learning techniques used to estimate it are explored. Evaluation of this controller shows considerable improvements in delays by comparison with current industry practices.

Impact Statement

Transport operators are investing considerably to research and develop intelligent infrastructure systems to take advantage of the emerging information and communications technology (ICT) to manage their operations more efficiently. In the rail industry, the aim is to get more out of the existing railway infrastructure to facilitate dealing with future anticipated demand without the need for building expensive new rail infrastructure, therefore resulting in a better value for investment.

In Great Britain, the current infrastructure manager Network Rail plans to invest millions of pounds to develop and deploy such systems, referring to this programme as the *Digital Railway*. One of the main projects in development by Digital Railway is Railway Traffic Management (RTM). This thesis has developed RTM algorithms which would help realise the vision of a Digital Railway. The main element of any RTM system is the underlying algorithms, and as such, the present research contributes to the foundation of a digital railway.

The adaptive railway traffic control system presented in this thesis is based on approximate dynamic programming (ADP) and uses reinforcement learning (RL) to explore the underlying process of railway traffic from real-time data to make estimation of long term performance. A growing and rich literature is available on RTM algorithms, though few consider data-based approaches to solve this complex optimisation problem. This thesis shows the potential of such approaches, which need to be further explored to realise a future where intelligent transport systems aid integrated and seamless transport.

The findings of this thesis have been published in the following journals:

- Taha Ghasempour and Benjamin Heydecker. Adaptive railway traffic control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 2019. https://doi.org/10.1016/j.trc.2019.04.002

- Taha Ghasempour, Gemma Nicholson, David Kirkwood, Taku Fujiyama, and Benjamin Heydecker. Distributed approximate dynamic control for traffic management of busy railway networks. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2019.2934083

The findings of this thesis have also been presented at the 23rd International Symposium on Transportation and Traffic Theory (ISTTT23).

Acknowledgements

First and foremost, I would like to thank my principal advisor, Professor Benjamin Heydecker, without whose guidance and continuous support the completion of this thesis would not have been possible. I particularly enjoyed our discussions and collaboration on our journal papers. His approach to supervising my research and his conduct in general taught me valuable life lessons which I am grateful for.

I am also grateful to my colleagues at the Centre for Transport Studies, Taku Fujiyama, Andy Chow and Fang Xu, and University of Birmingham colleagues, David Kirkwood and Gemma Nicholson, for their support and input on this thesis, and to the Rail Safety and Standards Board (RSSB) for the financial support provided to me through the 'DEDOTS' project.

I am truly glad to have met some of the most intelligent, interesting and impassioned people in my department at UCL. My fondest memories have been made with these beautiful people in UCL's Chadwick Building.

I would like to express gratitude to my partner Sara, who strongly supported and encouraged me during my long years of research.

Finally, I would like to deeply thank my parents, Shahnaz and Ali, who selflessly dedicated all of their life for the betterment of mine.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

"It is the mark of an educated person to be able to entertain a thought without accepting it." - Aristotle

Today's cities are becoming larger and ever more congested. Railways are now more popular than ever before as means of commuting to cities and passenger numbers are expected to grow in many countries around the world for the foreseeable future. With the increase in passenger numbers over the past decades, and projections that this trend is to continue, governments and railway undertakings are therefore eager to increase railway capacity and customer satisfaction while at the same time managing cost and energy usage. This can be achieved by exploiting information and communications technology in control of railway networks. The reason behind this drive is to add more capacity and reliability into the railway network by improved usage of current infrastructure and capabilities at affordable cost [1, 2].

Timetabling (or scheduling) and traffic management are two essential elements of railway operations. They allow for effective usage of railway resources including infrastructure, crew and rolling stocks. Timetables also inform the operators and passengers of the movement of trains. Because railway timetables are based on deterministic running of trains and their dwell times at stations, time margins are added to timetables to accommodate unforeseen variations. Due to the time taken to compute a new timetable, full recalculation

in response to disturbances is not possible and in current practice train dispatchers rely on their own experience to modify railway operations to recover delays or limit their propagation onto other train services. However, the efficiency and effectiveness of these actions is often unknown. In such cases deploying automatic re-scheduling tools has been shown to improve performance by limiting delay and returning operation to the planned timetable as quickly as possible.

Real-time regulation of railway traffic aims at ensuring safe, punctual and energy-efficient train operations. These Railway Traffic Management (RTM) systems anticipate future conflicts based on current speeds and positions of trains and provide suitable control measures. By using this real-time information to assess requirements and opportunities, the motion of trains can be controlled advantageously through real-time management of sequencing of trains and of acceleration and deceleration of individual trains. This will facilitate response to perturbations and other minor deviations from scheduled operation and to major disruptions to expedite recovery; in doing so it will support the operation of enhanced timetables that meet the increasing call on rail network capacity by passenger and freight operations.

1.1 Railway traffic control and re-scheduling

Railway control systems exist to ensure that trains can run safely and efficiently. The railway system is primarily concerned with the movement of passengers and goods; therefore, railway control systems are charged with the positional regulation of trains. Railway control systems oversee: i) longitudinal control of trains to maintain a safe distance between following trains, ii) crossing and switch control to regulate the movement of trains at junctions and where crossing a path which could be taken by another train, and iii) platform control at stations to guarantee safe train movement at stations.

The characteristics of rail transport differ from those of other transport modes in a number of ways. Safe operation of trains cannot rely solely on the reaction of train drivers because the speeds and low wheel-rail contact friction of trains result in long braking distances. The required starting point for braking from high speed is usually beyond drivers' line of sight

[3]; hence, railway control systems make drivers aware of the need for braking well before a driver can assess conditions downstream.

While dispatcher's regulation of railway traffic, aims at ensuring safe, punctual and energy-efficient train operations, due to the time taken to compute a new timetable in the presence of perturbations, train dispatchers rely on their own experience to provide modifications to the live railway timetable to recover deviations from the working timetable, while the efficiency of this is often unknown.

Real-time re-scheduling introduces the process of varying the timetable by considering the current position and speed of trains, and their delays. Therefore, a real-time rescheduling system is equivalent to a dispatching support system which aids the railway operators to reduce delays and increase efficiency of operations. Such a system would be designed to re-schedule train movements in real-time following perturbations and, according to D'Ariano [4], typically contains the following components:

- Conflict anticipation: Given the current infrastructure status, timetable, rolling stock information, and the position and speed of each train, find potential conflicting train services in a pre-established period of traffic prediction,

- Conflict resolution: Given the actual train delays, propose how to resolve anticipated conflicts based on the most suitable and robust dispatching options to minimise delays and, if required, energy consumption.

The critical factors for a good traffic management system can be summarised as use of limited time to compute solutions, the ability to recover train delays, and a fast feedback for rail operations. Hence, in the practical implementation of this system, important issues are to compute appropriate dispatching measures and to co-ordinate the actions at network level. This is a feasibility problem for which deadlock-free and conflict-free solutions have to be computed within a time appropriate for rail operations.

1.2 Solving the re-scheduling problem

Real-time RTM involves making decisions based on known information of the railway network, together with available information from real-time observations. Such problems are known as sequential decision problems, where decisions have to be made for the immediate future, but subsequent decisions are better made, or informed, once future information becomes available.

Given the diversity of applications that can be modelled as a sequential decision problem, it should not be surprising that there are more than one way to approach solving this optimisation problem. All approaches reported in the literature to solve these problems can be considered as one of: i) exact algorithms, that evaluate all of the search space to find an optimum solution, and ii) approximate algorithms, that trade optimality of exact solutions for efficiency in time by obtaining near-optimal solutions at relatively low computational cost.

Many optimisation algorithms have been investigated for re-scheduling. Majority have been applied due to their success in solving comparable problems. The successful mathematical approaches already applied to railway re-scheduling can be divided into:

- Graph-based algorithms, where the solution space is organised to allow for a structured way of searching for solutions [5, 4].
- Mixed-integer linear programmes, which are linear programs where some variables are required to take integer values [6].
- Evolutionary algorithms, where solutions are found through iterative improvements [7].

It is apparent that approximate approaches, such as evolutionary algorithms, have been shown to be capable of solving the re-scheduling problem.

An approach that has been reported to be effective in solving control problems [8], and specifically traffic control problems [9, 10], in recent decades, and not systematically investigated for railway re-scheduling, is the approximate form of dynamic programming. In

this thesis, we focus on this unexplored approach that has been effectively applied to similar applications.

1.3 Approximate dynamic programming

A popular approach used to model sequential decision problems is dynamic programming (DP) [11], which provide exact solutions for optimisation over time. DP decomposes a sequential problem into a series of sub-problems with discrete time-steps between them. At each time-step, the system is characterised by a number of state variables that specify the sub-problem. Computation routine of DP involves construction of the *value function* that associates the optimised future value with each state. Therefore, the value function represents the long term costs/rewards associated with optimal future actions. The property required to solve a DP is referred to as the *principle of optimality*, which states that knowledge of the current state of the system conveys all the information necessary about its previous behaviour for determining the optimal policy henceforth. Under this principle, results obtained from DP are globaly optimal.

Even though DP characterises an elegant way of representing an optimisation problem, it suffers from the *curses of dimensionality* [12]. This refers to the computational demand involved in calculations of DP which are exponential to the size of each of the state space, information space and decision space. Therefore DP is not practical for operational use where large state and action spaces exist.

To address difficulties associated with DP, the approximate dynamic programme (ADP) approach to decision-making has been developed. ADP is a variant of DP which uses approximation to evaluate future states subject to subsequent optimal control decisions. It is often used when the foresight of a DP is needed but the problem is too complicated for a DP strategy to be able to tackle it in an efficient and timely manner.

ADP is able to model complex problems with highly stochastic nature. Many data-based control approaches have also been developed based on DP principles in the recent decades. This means it is now possible to model complex sequential problems as DP with integrated

data-based components and solve them by using real-time data to learn the underlying patterns of complex operations allowing for development of highly adaptive systems.

The foundation of ADP is based on an algorithmic strategy that steps forward through time to calculate near-optimum decisions. To do so, ADP approximates the optimal performance that could be achieved from future states, and plans accordingly. In this forward process, the focus is on the near-future for which information is more reliable and decisions imminent. In this study we focus on an approximation to the value function which is used to evaluate the future consequence of control decisions. This entails optimisaton of parameter values that lead to a sufficiently accurate approximation to the optimal value function. To achieve this, the machine learning method of *reinforcement learning* can be incorporated for computing parameters in real-time to update approximations according to each observation of state transition.

1.4 Reinforcement learning

Reinforcement learning (RL) is a method of machine learning, that is a computational approach to learning from interaction with the control environment, similar to how intelligent beings learn to perform tasks. RL methods are generally used in practical domains when a model of the system is not available or the dynamics of the system are too complicated to be captured analytically. In RL, a system that is learning from a control process is a learning *agent*, which seeks to improve the performance policy for each state, solely by means of interacting with the environment. In the most challenging cases of control, actions may affect not only the immediate outcome but also the next outcome and, through that, all subsequent outcomes. These two characteristics (trial-and-error search and delayed reward) are the two most important distinguishing features of reinforcement learning [13].

In many cases in the literature [14] ADP and RL are used interchangeably to refer to the same learning based algorithmic strategies. This is because RL relies on the formulae of DP to estimate future values of performance as well as the immediate ones, therefore, it is closely related to ADP where it can be employed to learn with parameterised approxi-

mations to address the classical *curses of dimensionality* in DP. Indeed, Tsitsiklis and Van Roy [15] proved that for linear function approximation, under certain assumptions, this approximation process for the parameters converges to the unique optimal parameter values with probability 1. As for nonlinear approximation, general conditions for convergence are yet to be established. As RL and ADP share important fundamental characteristics, RL is used in this thesis to develop ADP for railway traffic management.

1.5 Research objectives

A growing literature is available on real-time RTM [16, 17, 18] which has developed the theoretical effectiveness and benefits of using such tools. However, a review of the available literature on RTM reveals that comparatively few industrial prototypes of these systems have been implemented in practice [19, 20, 21].

One significant difficulty with the approaches to real-time train re-scheduling is that they are heavily affected by the size of problem instances. The complicated nature of railway operations means that there are a large number of: i) possible states; ii) feasible actions which can be taken; and iii) possible outcomes of making each decision. This means that for complex congested railway networks calculating optimal solutions could be computationally intractable due to the number of cases to be considered, which renders RTM for large railway networks too inefficient to implement. Furthermore, most re-scheduling approaches in the literature assume optimisation in an ideal environment and report results from testing in simulations configured accordingly. However, the railway environment has a highly stochastic nature in which efficiency and robustness of re-scheduling controls are the essence of reliable operations.

It is therefore desirable to build upon the existing re-scheduling methods to develop frameworks in which optimisation can be achieved reliably and in a timely manner as this represent the practical requirements of railway operations closely. This will increase the chances of effective implementation of real-time traffic management systems in practice.

Given the robustness of ADP in tackling optimisation and control problems, as is presented in the literature, the purpose of this investigation is to develop an RTM framework that is adaptive (adjusts the parameters that control responses), based on ADP with distinct data-based learning techniques, that optimises railway traffic by controlling train sequences at critical points (such as junctions, merges and crossings). The framework that is developed for this is evaluated, by computer simulation, to investigate the effectiveness of ADP in tackling railway re-scheduling for large and dynamic networks, and also in a stochastic environment to investigate the performance of our adaptive approach under uncertainties present in practical operating environments.

1.6 Outline of this thesis

In the remainder of this thesis, railway control systems and the state of the art in real-time railway re-scheduling are reviewed in Chapter 2. Chapter 3 introduces fundamentals of ADP, outlines how it reduces computational burden compared to DP, and develops learning techniques based on reinforcement learning, hence forming the methodology of the framework developed in this thesis. Chapter 4 presents the proposed RTM framework by combining knowledge and methods from Chapters 2 and 3. Chapter 5 evaluates the adaptive framework of Chapter 4 in simulations representing the stochastic nature of rail operations, and on a large and dynamic network. Finally, a summary of this thesis and concluding remarks on future research is presented in Chapter 6. Figure 1.1 presents the objectives and relationships of the aforementioned Chapters.

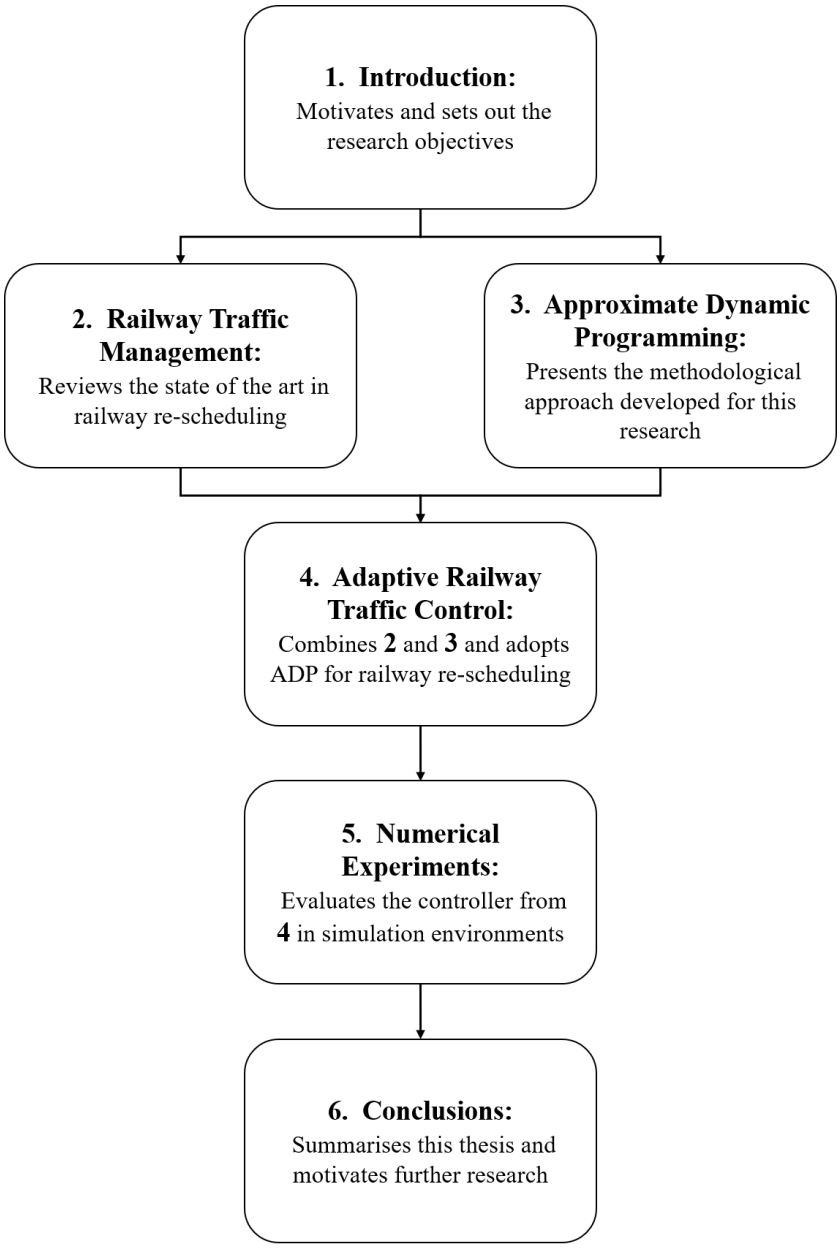


Figure 1.1: Outline of this thesis with each chapter's objective.

Chapter 2

Railway Traffic Management

"The knowledge of anything is not acquired or complete unless it is known by its causes." - Ibn Sina

To ensure safe railway operations, movement of trains traveling on the same tracks are regulated using railway control systems. These systems ensure that a safe distance separates all trains from others they are following to give train drivers the required distance to bring a train to a halt if necessary. This limits capacity utilisation but is essential in terms of safety. To maximise operational efficiency while respecting safety rules of railway control systems, railway timetables are produced in a complicated and exhaustive process that may take months. Unfortunately, unrecoverable deviations from the timetables may result in poor performance and therefore, in such instances, a revised timetable (or schedule) could improve performance.

With the emergence of new control systems that are intertwined with advanced information and communications technology, real-time data from the railway network can be exploited readily to re-schedule plans that would result in an optimised performance.

In this Chapter we review current and future railway control systems to asses opportunities for real-time traffic management, review current state of the art in real-time railway re-scheduling approaches, and examine the prospects of approximate dynamic programming methods for re-scheduling.

2.1 Railway control systems

Railway control systems ensure safe and efficient operations, so as to satisfy the expectations of customers, regulatory authorities and the railway operators [22, 23]. As transport systems are mainly concerned with the movement of passengers and goods, railway control systems are tasked with positional control of trains on tracks. Woodland [3] summarises the essential purpose of any train control system as:

- Longitudinal control: maintain a safe distance between following trains and regulate the passage of trains according to the service density and speed required, accounting for the planned schedule;
- Crossing and switch control: safeguard the movement of trains at junctions, and where crossing a path, which could be taken by another train;
- Platform control: control train movement between, and at, stations.

This clearly makes railway control systems an integral part of the railway system, and as they ensure safe operations by controlling movement of trains, they inherently affect capacity on the network and speed of trains. This places conflicting requirements upon the control system: those of safety and those of operational capacity.

Traditionally, the objective of train control systems has been centred around safety, and other considerations such as capacity have been promoted as secondary; the rational being, an unsafe transport mode would not be practical and not able to compete with other transport modes.

The major difference in rail transport compared to other transport modes is that train movements are restricted to one degree of freedom and trains only follow the direction of rails. Given the high speeds of trains and the low coefficient of friction between rails and train wheels, giving rise to long braking distances, it is in most cases unsafe to rely on train drivers' reactions to bring trains to a halt if another train is occupying their path. Therefore, railway control systems have been developed to assist train drivers and inform them well in advance of a stopping point to start applying the brakes.

Furthermore, fixed railway infrastructure and the requirement to maintain a safe longitudinal separation between trains, make planning track access (or scheduling) for trains of utmost importance in terms of railway operations. The aim of railway scheduling is therefore to ensure optimal usage of the available rail infrastructure.

In the remainder of this Section, current and planned railway control systems are introduced to assess requirements and opportunities for prospective railway traffic management systems.

2.1.1 Fixed block colour light signalling

Current railway control systems are still mostly based on *fixed block signalling*, in which all tracks are divided into a series of fixed locations called *blocks* [24]. The operational principles of these systems are [3]:

- A train cannot be authorised to enter a block if it is occupied, and
- The distance separating a train from the next one downstream (ahead) must always be greater than the braking distance required for the rear train to stop safely.

According to these principles, knowledge of trains' positions at all times is critical, and therefore train detection systems are used. In fixed block colour light signalling instead of reporting the exact position of trains, unlike other positioning systems, occupancy of blocks is automatically monitored; if a block is unoccupied, then another train can use this block. In one widely used detection method, the control system detects the presence of a train in blocks automatically and at all times using *track circuits*.

Track circuits are the earliest, yet still most widely adopted train detection sensors. The principles governing track circuits are simple which has resulted in their popularity. In its simplest form, track sections are insulated from adjutant sections and a feed provides a voltage through the tracks to a relay at the other end of the section. If the section is empty, electric current can circulate in the rails and the relay without interruption, energising the relay, as long as the section is free, and indicating an unoccupied track section. In the event

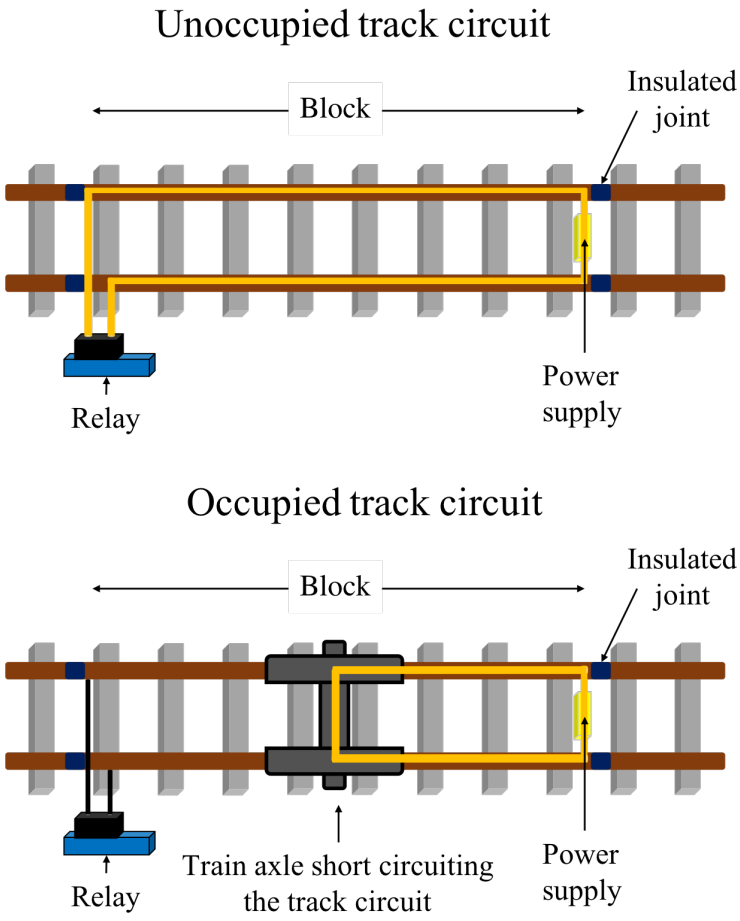


Figure 2.1: Working principles of unoccupied and occupied track circuits.

of presence of a train in the section, the axle of the train will interrupt the electric current by providing a path with lower resistance between rails, causing a short circuit. This causes the relay to drop due to the lack of current and this indicates an occupied track section [25]. This working principle of track circuits is illustrated in Figure 2.1. It has to be noted that other forms of train detection also exist, *e.g.* axle counters, where number of axles going into and out of blocks are counted and compared.

With the assistance of train detection sensors, train movements are controlled using traffic lights, similar to those on the road network, placed on the side or above the tracks. Train drivers are informed of occupancy status of tracks downstream by the colour of these signals.

2.1.1.1 Two-aspect signalling

The most basic form of colour light fixed block signalling is the *two-aspect signalling* system that at each instant shows one of *red* (stop) or *green* (proceed) aspect. Two-aspect signalling is normally employed on networks where train driver's sighting distance exceeds barking distance of trains, and therefore, drivers are able to sight and react according to the signals. In such cases, once a driver observes a red signal, they are required to bring the train to a halt before the front of the train reaches the signal. In the case of a green signal, drivers may proceed to the next block beyond the current signal.

As signals may fail and signalling systems are vulnerable to driver error, an additional safety distance, called an *overlap*, beyond the signal is required to be unoccupied for the signal to authorise movement into the block. The distance for overlap is usually calculated to present the worst case stopping distance under emergency braking plus a safety margin to allow for a safe distance to be present between trains in this worst case. In addition to the overlap, another additional safety distance exists, this time between the point where the driver should be able to sight the aspect of the signal, the *sighting point*, and the signal itself. This is termed the *sighting distance*, and it allows the driver to sight the signal and, in case of a red signal, bring the train to a halt safely using service braking rate before the front of the train reaches the signal.

Figure 2.2 [24] presents the layout of a two-aspect signalling system, and shows the mini-

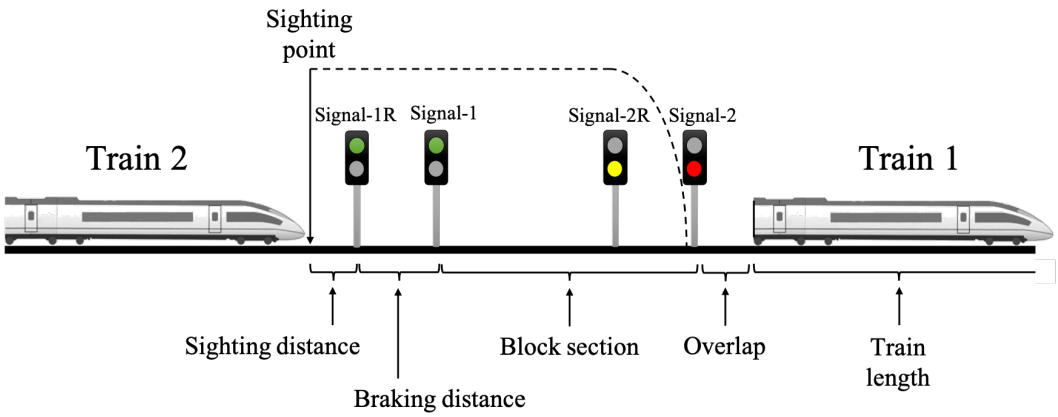


Figure 2.2: A two-aspect signalling system.

mum distance required between the front of two following trains to ensure safe operations. This safe separation distance is given as

$$H_2 = S + B + X_2 + O + L, \tag{2.1}$$

where

- H_2 is the two-aspect spacing distance,
- S is the sighting distance,
- B is the braking distance,
- X_2 is the signal separation distance or block length,
- O is the overlap length,
- L is the length of train 1.

In instances where, for any reason, the signal is not observable by the driver, a repeater signal is installed to inform the driver of the upcoming signal aspect; this is presented as "Signal 1R" and "Signal 2R" in Figure 2.2.

Two-aspect signalling systems are normally employed in underground (metro) systems. The reason being the relatively low speed of operations in underground systems which allows train drivers to safely respect signals as they come into sight. In cases where higher operational speeds are required, two-aspect signalling may not be practical.

2.1.1.2 Three-aspect signalling

To regulate train movements on mainlines, where high operational speeds are common place, three-aspect signalling offers improved practicality compared to two-aspect signalling. In this approach, signals are capable of displaying three aspects [26]:

- Green for proceed,

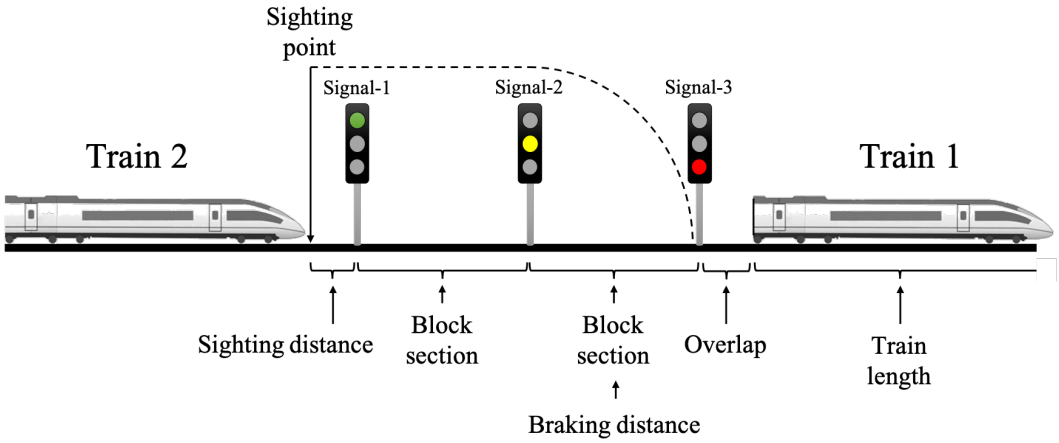


Figure 2.3: A three-aspect signalling system.

- Yellow for caution, and
- Red for stop.

The layout of a three-aspect signalling system is presented in Figure 2.3 [24]. In this setup, if a signal shows a green aspect, the driver can be sure of at least two empty blocks ahead, and therefore the train can proceed into the next section at maximum permitted speed. If a yellow aspect is displayed, the driver is required to start braking as this means the next signal is currently displaying a red aspect.

The minimum safe spacing distance in a three-aspect signalling system, H_3, is therefore calculated by

$$H_3 = S + 2X_3 + O + L\,. \tag{2.2}$$

The best achievable headway under the three-aspect system will be when signal spacing is equal to the calculated braking distance, which is not an improvement on what is achievable by use of two-aspects signalling.

2.1.1.3 Four-aspect signalling

To increase line capacity compared to three-aspect signalling, four-aspect signalling can be employed [27]. In this approach the signals are brought closer together, meaning that blocks are now shorter than the braking distance of trains. The warning that the train is

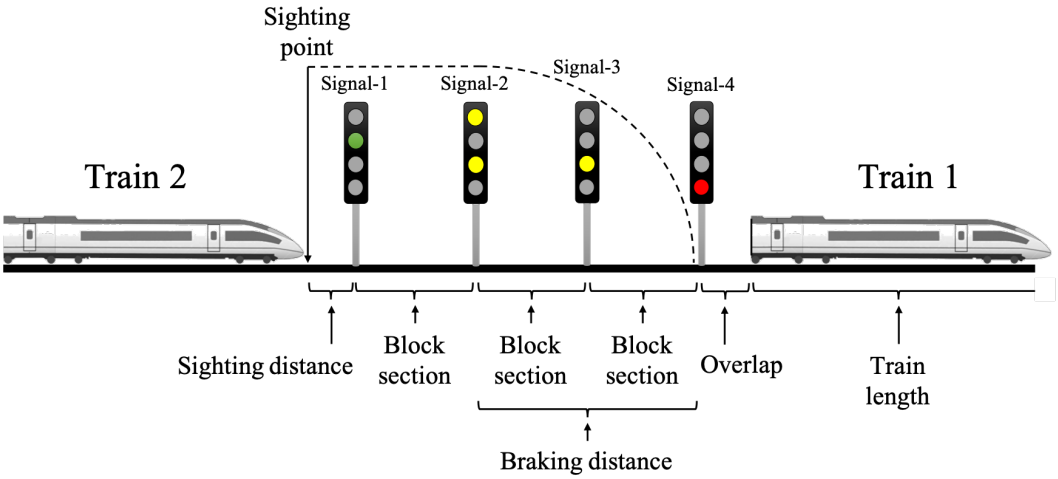


Figure 2.4: A four-aspect signalling system.

approaching a red signal is still delivered to drivers at a safe position, however this is not delivered by the first signal before the red signal, but by the one before that. In a four-aspect signalling system, the aspects are:

- Green for proceed,
- Double-yellow for preliminary caution,
- Yellow for caution, and
- Red for stop.

The sequence of these aspects is presented in Figure 2.4 [24].

Under these conditions, drivers are required to start braking after a double-yellow signal. The safe separation distance is then calculated by:

$$H_4 = S + 3X_4 + O + L\,. \tag{2.3}$$

If the average signal spacing is equal to half of the calculated braking distance, then the resulting headways would present an improvement on what is possible under two and three-aspect signalling systems. A further advantage on three-aspect signalling is that signals do not need to be placed with the same separation distance, as long as the distance between

a double-yellow and a red signal allow trains to safely brake to a halt. This approach requires more signals to be installed, therefore this system is more expensive than the others discussed so far in this Section.

2.1.1.4 Five or more aspect signalling

The same principle can be applied to adopt any number of aspects for railway signalling. A generalised headway distance equation for three or more aspects can be presented as:

$$H_n = S + (n-1)X_n + O + L \text{ , where } n \geq 3 \text{ .} \tag{2.4}$$

Railway operators use signalling systems of four or less aspects, due to the cost associated with five or more aspect signalling systems. Furthermore, train drivers would not be able to observe and react to signal aspects satisfactorily in systems with high aspects. Therefore, the most common signalling system on mainlines are four-aspect signalling systems which offer a good balance between capacity utilisation, cost, and ease of use for drivers [3].

2.1.2 Moving-block signalling

Fixed block signalling, significantly limits capacity on railway networks, due to the fact that blocks are fixed infrastructures and entry into blocks is only permitted once the block and its overlap are unoccupied. In practice this translates into trains accelerating toward signals,

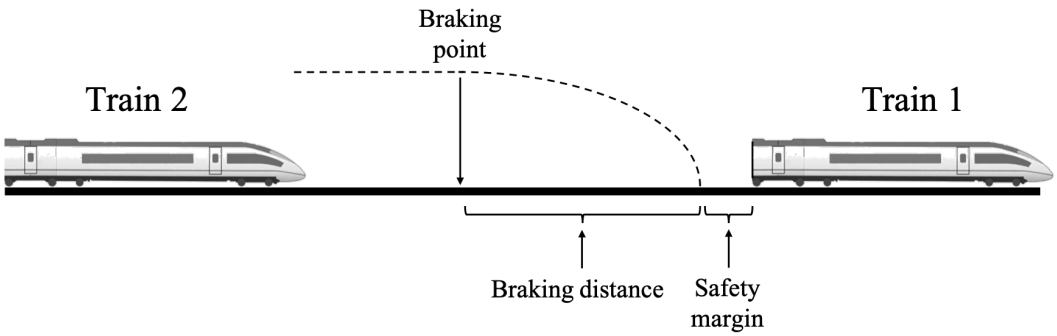


Figure 2.5: A Moving-Block signalling system.

then stopping for them if a red aspect is displayed, and start accelerating from a speed of zero once the signal turns green. Trains are inherently heavy, and accelerating from a zero speed requires significant power and takes a long time, therefore having trains constantly stopping and starting to move is clearly time consuming and not efficient which further reduces capacity utilisation.

A proposed system to improve on fixed block signalling is to abandon the concept of fixed blocks on the rail network altogether. This idea, which was first introduced in 1938 [3], requires trains separation to be reduced to running sufficiently far behind the leading train as to respect a safe braking distance, which is dependent on the relative speeds and characteristic of the following train. This system is commonly referred to as *moving-block* control and requires temporally continuous monitoring of speed, position and control of trains.

The concept of moving-block train control is one which is applicable to fleeting or following train moves. In this concept, the movement authority limit, which specifies the distance that a train is permitted to travel, of a following train is extended nearly constantly to the location of the rear of the leading train. As the name implies, in some cases the limit of authority is not a fixed signal location, but can move along the track.

Apart from the infrastructure cost savings associated with moving-block signalling compared with fixed-block signalling, the most significant improvement in terms of capacity is achieved by adopting a varied braking distance $B(v)$ which is specific to instant train speeds v and therefore, depending on the capabilities of trains, the braking distance can be adjusted to the instantaneous safety requirements of operations. This allows trains to safely close up at lower speeds, so that during disruption or on the approaches to stations, headway is not lost because trains are running slowly through sections spaced for higher speeds.

Figure 2.5 [24] presents a moving-block signalling systems, where trains are controlled to stop at a safe headway distance behind the leading train, instead of at fixed location signals. This has the effect of allowing trains to travel closer together, *e.g.* during disruptions, instead of stopping at a location designed for higher speeds trains can follow leading trains more closely.

Therefore, the minimum separation of trains in a moving-block signalling system, can be calculated as:

$$H_m = B(v) + O + L \,. \tag{2.5}$$

2.1.3 Automatic Train Control

Automatic Train Control (ATC) is a generic term to describe systems that reduce the amount of human involvement in the operation of trains. ATC has been introduced in railway systems with the primary goal of improving safety, although other benefits resulting from automation could include reduced staffing costs and more consistent regulation of train services.

As railways have grown in complexity and traffic, ATC has become more common place to minimise human errors in operations. As automatic control allows for more sophisticated control strategies, ATC systems can be employed to [24]:

- Increase safety,
- Reduce energy consumption,
- Reduce the amount of track-side equipment, thereby decreasing cost,
- Improve passenger comfort, and
- Improve the performance of railway control systems.

Adopting ATC on underground systems is more straightforward compared to mainline railway lines, as the similar train mechanical characteristics and stopping patterns on underground lines, make automation less complicated. Mainline railway lines are more difficult to automate fully because, they include more complicated mixed traffic scenarios, and therefore, mainline railways tend to employ limited ATC.

ATC can be separated into three functions:

- Automatic Train Protection (ATP),
- Automatic Train Operation (ATO), and

- Automatic Train Supervision (ATS).

These will be discussed in turn.

2.1.3.1 Automatic Train Protection

Automatic Train Protection (ATP) is used to automatically activate train's braking systems in cases where the train speed exceeds that of the permitted speed-limit, or if trains pass red signals. Therefore, ATP is used to ensure safe driving of trains with respect to permitted speeds on the rail network [28].

To accomplish safe speeds, trains carry 'fail-safe' computers that monitor speeds while trains are moving. These calculations require the following data [26]:

- Current train speed,
- Train length,
- Train braking performance,
- Maximum permitted speed of train,
- Route data , which includes maximum line speeds, and
- 'Distance to go', which informs trains of the distance to the next stopping or slowing down position; data such as any Temporary Speed Restrictions (TSRs) can also be accounted for in this data.

The ATP system is therefore able to calculate continuously the maximum safe speed by constructing braking curves for the next stopping or slowing down position. If train speed exceeds this maximum safe speed then the following sequence of actions will be applied [3]:

- A warning will be displayed to the driver,
- If the speed continues to exceed the maximum safe speed, then service brakes will automatically be applied,

- If service braking is not able to reduce the speed as desired, then emergency brakes will automatically be applied to bring the train to a halt.

There are two main classes of ATP systems: continuous ATP and intermittent ATP. As the names suggest, continuous ATP is capable of receiving movement authorities continuously, while intermittent ATP can only receive movement authorities when the train passes a 'beacon'.

2.1.3.2 Automatic Train Operation

In principle, the braking curves calculated by ATP systems are how train drivers should drive to comply with the signalling systems and hence ensure safe operations on the rail network. Therefore, ATP can be extended to drive trains automatically; this is called Automatic Train Operation (ATO). The benefits of such a system include [26]:

- Cost savings associated with training and employing train drivers,
- Elimination of human errors in driving trains,
- Smooth, consistent and regulated accelerations, and
- Instant reaction to changing conditions on the line.

Many railways around the world, employ ATO and ATP systems concurrently on trains. Even though ATO is an extension of ATP, the systems are maintained separate as the functionality is different and ATP, being a 'fail-safe' system concerned with safe speeds, should be able to intervene in decisions made by ATO. Therefore, it is usual to install both ATP and ATO on trains, whereas ATP can be installed without ATO.

2.1.3.3 Automatic Train Supervision

Automatic Train Supervision (ATS) is a generic term for describing systems which reduce the reliance on human involvement in the central control functions of a railway [3]. While ATP and ATO are concerned with automating actions of human drivers, ATS is concerned

with automation of signallers' roles, and is therefore responsible for co-ordination of all individual train movements in accordance with operational requirements.

Adoption of ATS is generally expected to enhance operations by performing tasks such as [26]:

- Routing trains as efficiently as possible according to service patterns,
- Automatic re-scheduling of services in the event of perturbations,
- Relaying information regarding current conditions to trains' systems, and
- Regulation of traffic in case of minor deviations from the timetable.

ATS is therefore intended to improve performance and is not a vital control system, the failure of which would not directly affect railway safety.

2.1.4 European Train Control System

The European Train Control System (ETCS) is the ATP system component of the European Rail Traffic Management System (ERTMS), a European Union led system to facilitate the interoperability of a trans-European rail system. ERTMS aims to deploy a unified traffic management system for Europe which utilises modern communication technologies to develop a new and more efficient signalling system [29]. Other similar systems also exist around the world, such as the Chinese Train Control System (CTCS).

As well as ETCS, ERTMS also includes the Global System for Mobile Communications-Railway (GSM-R) which is a radio-based system providing communications between trains and control centres, and the European Traffic Management Layer (ETML) which is a concept to optimise railway operations through improved management of train running to maximise the potential of a given layout and to reduce scheduling conflicts [29].

Railway operators around the world are moving toward deploying the ideas and concepts developed in systems similar to ERTMS, and have made commitments to use these systems to control railway traffic in the near future. As such, ERTMS presents the direction of

developments of future railway control systems, and so in the remainder of this Section, the control component of ERTMS, *i.e.* ETCS, will be presented in more detail. ETCS has been divided into different functional levels, described by the ERTMS Programme Team (and adopted in this thesis) as levels 1, 2 and 3, which are considered in turn.

2.1.4.1 ETCS - Level 1

ETCS-Level 1 is designed as an overlay on the conventional signalling systems presented in Section 2.1.1. In this system, the movement of trains are dictated by the conventional signalling system and its trackside equipments, such as the signals and track-circuits.
Operation of Level 1 requires use of intermittent transmissions from *Eurobalises*. These are beacons that are placed near signals on the rail network and transmit data to trains as they pass over them. There are two variance of Eurobalises:

- Fixed-data balise: transmits stored data to train's on-board antenna, and
- Variable-data balise: transmits variable data to train's on-board antenna and is controlled by a line-side electronic unit (LEU) which is connected via a permanently attached cable.

As a train passes over a balise, the on-board antenna activates the transpondere by emitting a low powered electromagnetic signal; this prompts the balise to transmit its information using the power from the signal, and depending on the size of data, transmits can be made at speeds of up to 500 km/h [30]. Using data transmitted via balises, trains' on-board systems are able to calculate braking points and speed profiles required for safe operations, and display the Limit of Movement Authority (LMA) information to the driver via the Driver Machine Interface (DMI). Figure 2.6 [29] shows an illustration of the ETCS DMI and the other in-cab instruments.
The train-borne ETCS system, using the absolute location information from balise mes-

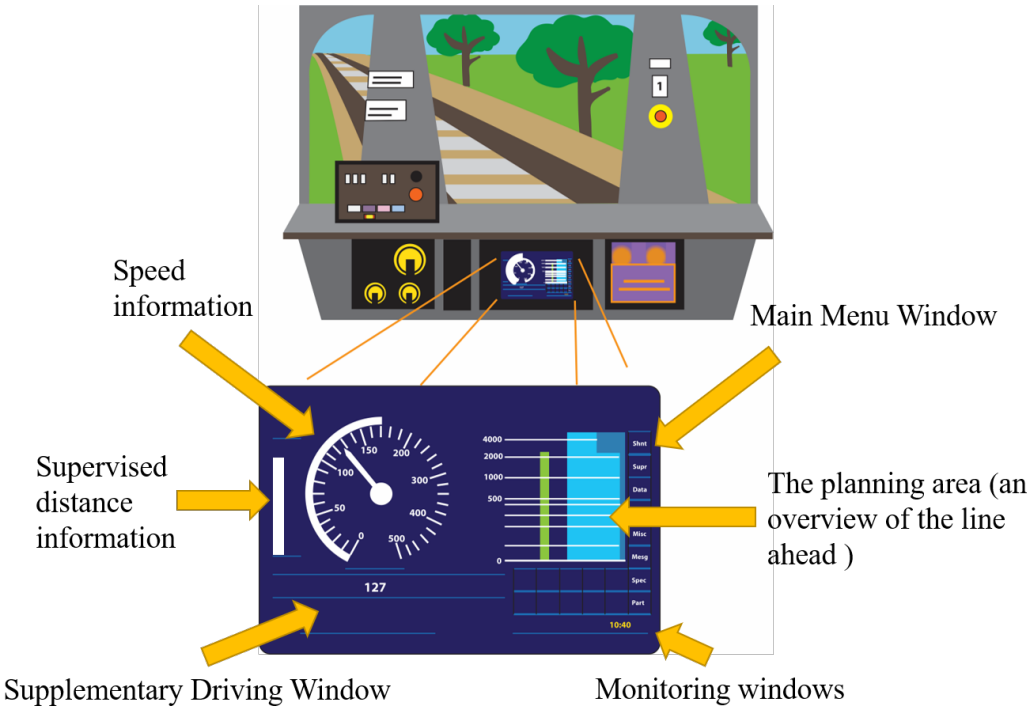


Figure 2.6: ETCS DMI and the in-cab instruments.

sages and an odometry system, determines speed and location of the train and therefore is able to provide continuous speed supervision and protects against overrun of the movement authority. In the event the movement authority is exceeded, emergency brakes are applied. In ETCS-Level 1, given that ATP is in use and monitors speeds based on the calculation of braking curve, the movement authority can only be updated once trains pass over a balise. As ATP is a fail-safe system, an infill balise is placed a further distance before each signal

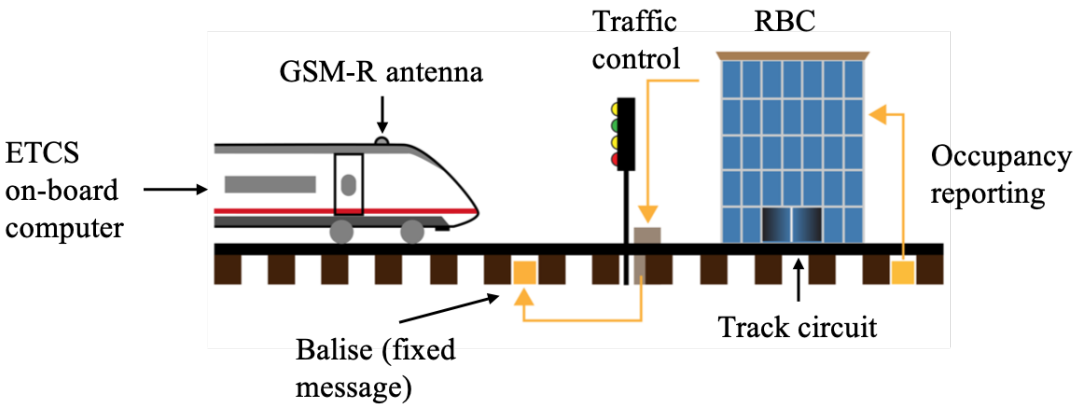


Figure 2.7: The working principle of ETCS - Level 1.

to provide information on block occupancies downstream as early as possible so that the driver is able to adjust train's speed without ATP overriding this decision. Figure 2.7 [29] presents the principles of ETCS-Level 1.

2.1.4.2 ETCS - Level 2

Similar to Level 1, ETCS-Level 2 can be used as an overlay on the conventional railway signalling systems. In this setup, train detection and separation functions are based on principles of fixed block colour light signalling systems, with intermittent balises mainly providing location referencing. The main difference with Level 1 is the use of continuous wireless communication with the traffic control centre by GSM-R. Movement authorities are then transmitted to trains by radio from the central interlocking rather than the signals. Therefore, under Level 2, a train does not have to wait until it passes over a balise to get a new movement authority, it can receive them by radio at any time that sufficient blocks are cleared ahead of the train [31]. The in-cab DMI will then display the movement authority beyond what may otherwise be observed from the signals, and when appropriate authorise operation at higher speeds. Figure 2.8 [29] presents the principles of ETCS-Level 2.

Given the principles of ETCS-Level 2, this system can operate purely based on the in-cab signalling provided by the DMI and without needing line-side signals. The main advantages are, reduced cost of installment and maintenance of traffic signals, and a more responsive

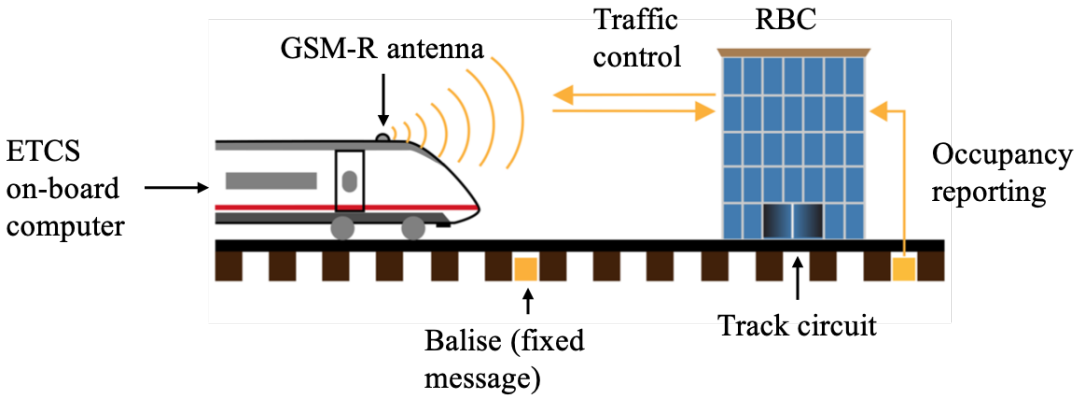


Figure 2.8: The working principle of ETCS - Level 2.

traffic control system that issues movement authorities immediately by radio as and when they can be issued. Drivers can therefore adjust train speeds appropriately even if signal sighting points have not been reached.

2.1.4.3 ETCS - Level 3

Unlike Level 1 and 2, ETCS - Level 3 does not rely on conventional fixed block signalling's train detection. In a Level 3 system, train location and mitigation of the risks involved in the loss of train integrity are performed by a track-side *Radio Block Centre* (RBC), in conjunction with the trains themselves [31]. According to Level 3, all trains transmit their position and train integrity reports to the RBC, and the RBC in return, based on the position of all trains, issues movement authorities back to the trains. These information is then displayed on the in-cab DMI. Figure 2.9 [29] presents the principles of ETCS-Level 3.
Thus far all the different functional levels of ETCS presented, comply with the principles set out by fixed block signalling; however, as Level 3 does not rely on fixed train detection systems, unlike Level 2, it is possible to implement it as a moving block signalling system. This may allow for better capacity utilisation, as trains could travel closer together. Other benefits could include: improved perturbation recovery and the ability to have bi-directional movement on all lines [3].

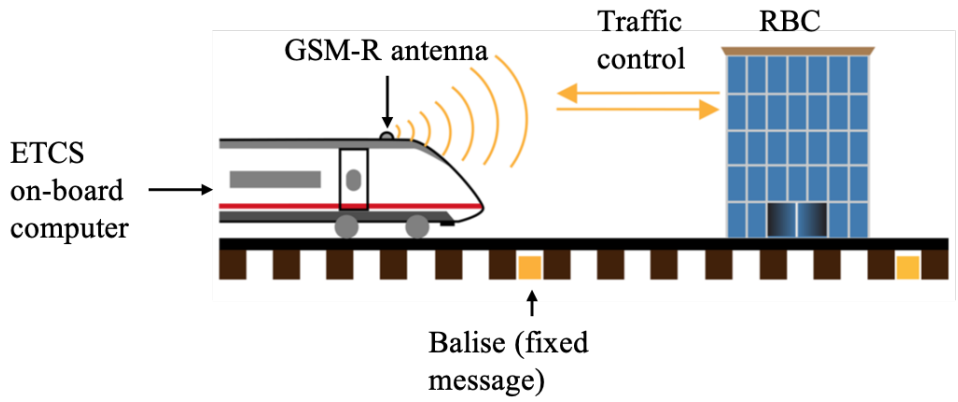


Figure 2.9: The working principle of ETCS - Level 3.

2.2 Real-time railway re-scheduling

Given the safety restrictions put in place by railway control systems and the complex nature of operations, railway operators produce timetables to use the system capacity at their disposal more effectively. Timetabling is the process of constructing a set of planned running schedule for trains, which includes times of arrival or departure at specific locations along the network [32]. The aim is to produce a schedule that is conflict free, can cope with minor deviations from the timetable in case of unforeseen events, passengers can use to plan for using services, and crew can use to operate services.

To construct such a schedule, detailed calculations of train running times between specified timing points on the network, *e.g.* scheduled stops and junctions, are made based on the characteristics of railway infrastructure and rolling stock. Recovery times and margin times also need to be added on to train running times to cope with driving variations and minor delay incidents to make the timetables "robust" [33].

Even though timetables are planned in meticulous detail and produced in a time consuming process, in cases of unexpectedly long delays, they may not fully utilise the capacity. It is conceivable that in a delay situation, *e.g.* longer than expected dwell times, the time margins put in place to account for relatively small deviations from the timetable are not able to absorb the delays and therefore the whole planned schedule could be jeopardised, resulting in poor performance as the deviations propagate in the network and effect other services [34].

In such cases, producing a new schedule in real-time has been found to improve performance [4]. This process is referred to as *re-scheduling*, as a new schedule is produced because the nominal timetable (the schedule) can not assure optimal performance anymore. Railway re-scheduling approaches apply mathematical models that describe railway operations at various levels of detail, to anticipate potential train conflicts and compute control actions that minimise specific objective functions. Possible control actions can be varying the passing times of trains at specific locations or the order of trains at stations and shared sections of track, as well as changing their routes within certain areas.

In this thesis, different types of delays are distinguished according to the following definitions:

- *Primary delays* arise from deviations from normal operation traffic, e.g. longer than expected dwell times,
- *Consecutive delays* are ones caused to other trains from the primary delays, and
- *Total delays* are the sum of all primary and consecutive delays.

A growing literature is available on real-time railway re-scheduling [18, 17, 16] which has shown the effectiveness and benefits of using such tools in some particular circumstances; the state of the art in re-scheduling approaches is therefore reviewed in the remainder of this section.

2.2.1 Rule-based systems

To deal effectively with potentially disruptive deviations from the timetable in real-time, railway operators examine actions taken to successfully deal with certain conditions and process them into rules or decision support systems for re-scheduling.

Hyoudou et al. [35] developed an 'expert system' for Shinkansen train dispatching in Japan. The system has a problem solving architecture which is designed to emulate the problem solving processes of a dispatcher on the computer for generating a practical re-scheduling plan for disturbed railway traffic within a short time.

Fay [36] developed a rule based decision support system based on experts' knowledge in fuzzy rules of the 'IF-THEN' type. In this system, possible effects of different dispatching decisions are evaluated in parallel with simulation runs of the decisions based on a number of criteria. Similarly, Cheng et al. [37] developed a system based on fuzzy rules of the 'IF-THEN' type, with the rules established based on interviews of dispatchers.

In the United Kingdom, a system called the *Automatic Route Setting* (ARS) system has been widely used as a decision support system to aid dispatchers decision making. Kuhn

[38] described the ARS system, in which trains routes are automatically selected based on a set of predefined rules. ARS is implemented for single junctions and considers the relationship between the first trains from each approach to the junction.

2.2.2 Simulation-based approaches

Given that train movements based on traffic control systems, passenger movements, and the dispatching process of a railway system can be modeled, a robust simulation system can forecast the future state of railway operations [39]. This forecast can then be used to resolve possible conflicts on the railway network and hence used for re-scheduling.

Quan et al. [40] developed a simulation model to find deadlock-free schedules that minimise the travel time of trains, while abiding by the speed limits at each point on each route, and maintaining adequate headways between trains.

Weston et al. [41], similarly, developed a simulation model that, using detailed modelling of operations, would minimise the sum of the deviations from the scheduled arrival time and the number of missed passenger connections by repeatedly simulating forward in time and choosing the decision with the lowest value with respect to the objective function; this is tested against a relatively small control region. They note that, the method can be used to simulate over a larger region, however, the exponential growth in the number of decisions associated with a larger regions could overwhelm their framework.

Tomii et al. [42] use simulation models to minimise passenger's dissatisfaction. This is formulated as a constraint optimisation problem and a meta-heuristic is developed to solve this problem based on simulations. Extending this work, Kanai et al. [43] develop an algorithm that minimises passengers' dissatisfaction. The algorithm is a combination of simulation and optimisation. The simulation part consists of a train traffic simulator and a passenger flow simulator which work in parallel. The train traffic simulator forecasts future train diagrams considering the dynamic interaction between trains and passengers. The passenger flow simulator traces behaviour of all passengers and calculates how many passengers alight and board at each station. Passengers' dissatisfaction is also estimated

from the results of the passenger flow simulation. A tabu search algorithm is then use to optimise the objective function.

Luethi et al. [44] proposed a method that reduces buffer times without impacting schedule reliability, by anticipating delays in simulations and applying their method. They note close following of plans by trains to be a factor in improving performance.

2.2.3 Mathematical models

Mathematical models for real-time traffic control include those focusing on energy-efficient train running by means of trajectory optimisation [45, 46, 47] and train re-scheduling for reordering, retiming and/or rerouting of trains to optimise specific performance objectives [48, 6].

Varying levels of detail can be used to model the railway infrastructure, we can distinguish between microscopic [49] or macroscopic models [50]. The former considers detailed information on tracks, switches, signals, block sections (for which at most one train at a time is allowed) and the signalling and safety system. The latter instead depict event times associated to stations (i.e. departures and arrivals) and other points on the network.

The most widely reported mathematical methods for re-scheduling are based on Mixed-Integer Linear Programming (MILP) and Alternative-Graph representation which are discussed in the remainder of this section. Other methods such as stochastic programming, not strictly related to these categories are also discussed.

2.2.3.1 Stochastic methods

Among the few that have investigated and tested RTM methods in stochastic environments, Meng and Zhou [51] employed a macroscopic stochastic programming model to account for the stochastic nature of railway traffic to solve a single-track train dispatching problem under uncertain running times and capacity loss durations. The objective function adopted for this work was to minimise weighted combination of penalties for earliness and

lateness of trains. Quaglietta et al. [52] also tested a re-scheduling approach, formulated using Alternative Graphs (discussed later in this section), under uncertainty by employing a Monte-Carlo scheme.

2.2.3.2 MILP approaches

Adenso-Diaz et al. [53] formulated the real-time timetable and rolling stock re-scheduling problem in cases of large delays of one or more trains as a MILP, which are a variant on linear programming where a problem may be defined as the problem of optimising a linear function subject to linear constraints and some of the variables are integers, while other variables can be non-integers. Because the problem is hard to solve using available MILP solvers, a heuristic was developed to produce feasible solutions that approximately maximise the number of passengers transported within the planning period. The model was implemented in practice and resulting performance was reported to be satisfactory.

Törnquist and Persson [6] also formulated the rescheduling problem as a MILP model which considers reordering and rerouting of trains with the objective of minimising train delays; Törnquist [54] extends this work by presenting a heuristic approach to solve the same problem, and reports good solutions that are produced fast. Building on these works, Josyula et al. [55] represent the solution space of the re-scheduling problem as a binary tree and developed a sequential heuristic algorithm to tackle the problem.

Other instances of MILP formulation for rescheduling were formulated by Pellegrini et al. [56, 57] with the objective of minimising consecutive delays. They report: because integer programming problems are hard to solve in many cases, the computation time of problems increase dramatically with the size of the system.

Indeed, solving the re-scheduling problem using MILP has been found to be NP-hard [58, 59, 60], and therefore normally heuristics are developed to solve the problem based on MILP.

Corman et al. [61] use MILP to solve the re-scheduling problem by also taking into account minimisation of passengers' discomfort therefore integrating train scheduling and delay

management into one. This microscopic delay management (MDM) problem is modeled as a MILP that can be solved to optimality by commercial solvers for small-size instances. Three heuristic procedures are developed, based on decomposing the problem into a train scheduling problem and a passenger routing problem.

2.2.3.3 Alternative-Graph approaches

The other widely reported approach for re-scheduling is based on the Alternative-Graph (AG) representation. AG is a discrete optimisation tool which can be used to model re-scheduling problems with no-wait and no-store constraints which can be applied to job shops, formulated by Mascis and Pacciarelli [62]. Several works are reported in the literature using the AG model; among them D'Ariano et al. [63] proposed a branch-and-bound algorithm for re-scheduling trains in real-time, where block sections are characterised as machines and trains characterised as jobs in the formulation, therefore providing a microscopic view of the railway network under consideration. This formulation allows an efficient presentation of a railway network safety constraints (i.e. at any time, only one train occupies a block section).

Building on their previous work D'Ariano et al. [64] adopted a blocking time model to evaluate the feasibility of headways between consecutive trains, and train speed profiles are updated considering preceding signal aspects. Mazzarello and Ottaviani [65] similarly use the AG formulation to produce a conflict-free schedule for trains; they also compute train speed profiles to deliver the produced schedules to minimise train energy consumption. AG is also used to formulate the macroscopic model of railway re-scheduling by Kecman et al. [66].

Mannino and Mascis [19] developed an exact branch-and-bound algorithm for re-scheduling. The objective is to minimise the deviations of the actual schedule from the original plan together with a measure of the costs due to violating regularity (i.e. varying headways between consecutive departures). The results were compared to those obtained by human dispatchers and is reported to have improved punctuality and regularity by up to

8%.

Corman et al. [67, 68] extended the work of D'Ariano et al. [63] to include local re-routing of trains with the objective of minimising the maximum consecutive delay. Corman et al. [69, 70] extended their work by introducing a controller that co-ordinates multiple local areas to control large networks. For a limited number of areas, their framework is reported to perform well, but as the planning horizon is extended and the number of local areas is increased, good solutions could not be guaranteed.

Most of the approaches in the literature are designed as centralised systems to be implemented as open loop control. In open loop rescheduling the control measures are computed and implemented once and for all without monitoring of effects and responses, hence, assuming a perfect knowledge of future traffic states. This implies measuring only once the traffic status with the assumption that there will be no discrepancy between predicted and actual traffic conditions over time [71].

Distributed railway traffic management has also been described in the literature; Strotmann [72] presented a two-level approach for rescheduling trains between multiple areas. The lower level makes optimal solution in each local area while the upper level checks the feasibility and consistency of solutions in the neighbouring areas. A coordinator graph is used in such a way that any infeasible solution would result in additional constraints to the lower level solver.

Corman et al. [70] also developed ideas to solve independent and distributed re-scheduling with co-ordination between the distributed areas, and compared distributed and centralised approaches. They reported that the distributed system is computationally faster than the centralised system, and better performance in terms of the consecutive delays, especially for significant traffic disturbances, was achieved by the distributed system.

Corman and Quaglietta [73] Investigated closed loop control of traffic using the re-scheduling approaches described in this Section, where an open loop approach was compared with (i) repeated open loop, and (ii) closed loop (repeated open loop with knowledge of past events).

2.2.3.4 Other approaches

Brown et al. [74] applied both Simulated Annealing (SA) and Genetic Algorithm (GA) to determine train routes and train paths over a freight rail network, and found Simulated Annealing to perform better, but require higher computational effort.

Real-time implementation of GA for railway re-scheduling is introduced by Ho and Yeung [7], where a junction controller assigns a right-of-way sequence, with the objective of minimising the total delay and minimising the search time. Ho and Yeung [75] improve their work by investigating Tabu Search and SA for the same problem.

Ho et al. [5] developed and tested a traffic controller for railway junctions using dynamic programming (DP). Their method included approximations in the traffic flow model and optimisation process to improve computational speed. They reported that for isolated conflicts, where no new train enters during the rescheduling period, the controller produced a schedule with near optimal cost. In cases where new trains are expected to enter the control area at uncertain times, a probabilistic method was used which was found to achieve improvements of under 10% compared to *First-Come-First-Served* (FCFS). For a single converging junction, it was reported that the FCFS approach achieved results that were quite close to those of the optimal schedule when trains have the same speed characteristics.

Šemrov et al. [76] introduced a train re-scheduling method based on a reinforcement learning method, namely Q-learning. The evaluation of the approach is performed on simulations carried out on a practical railway network with bi-directional running. In this work the objective was to minimise the total delay, though the authors note that the objective function can be easily changed. The empirical results show that Q-learning leads to re-scheduling solutions that were at least equivalent and often superior to those of FCFS and random walk (*i.e.* selecting actions randomly).

Yin et al. [77] formulated a metro re-scheduling problem using a stochastic programming model that jointly optimises the delay experienced by passengers, their traveling time and train energy consumption. An algorithm based on approximate dynamic programming

(ADP) is then developed to solve the problem. To evaluate the algorithm, numerical experiments were implemented on a small simulated metro line and a practical instance of Beijing Metro to demonstrate the performance of the proposed approaches. They reported that the ADP approach reduced the passengers' delay by around 10%-30% compared with a commonly used heuristic method (*i.e.* if a train is delayed by 3 minutes at a station, the following trains are delayed by 3 minutes). They also reported that their computational time for generating a near-optimal solution by the ADP method was about 1-2 minutes.

Chen et al. [78] proposed an improved Differential Evolution algorithm for solving train re-scheduling problems in junction areas in the event of disturbances. The algorithm is reported to result in good performance.

Hirai et al. [79] proposed an algorithm for re-scheduling using a pattern processing system which surveys the train timetable and finds an appropriate running pattern that can be used by trains to prepare plans. Testing their method for heavy traffic disruption, caused by an incident requiring a train suspension for more than an hour, it is reported that the algorithm produced a re-scheduling plan suitable for practical use.

Kersbergen et al. [80] proposed a distributed model predictive control method to solve the re-scheduling problem for national railway networks. The method is evaluated using a model of the Dutch railway network. A method to split the entire network into subproblems is also developed, and the approach is evaluated using horizons of 30 to 75 minutes. The computational time to find optimal solutions for the entire network using long horizons is reported to be significant.

Very few industrial prototypes of these systems have been implemented [21, 19, 20]. It is difficult to evaluate, quantitatively, the benefits of re-scheduling systems in practice, as the improvements depend on the conditions encountered during experiments. A complete sensitivity analysis, to draw quantitative and general conclusions, of these systems is still lacking.

2.3 Practicality of ADP for re-scheduling

Railway re-scheduling is a resource planning and control problem. Problems sharing similarities to railway re-scheduling have been tackled using a variety of different techniques [81].

In recent decades, much attention has been paid to approximate dynamic programming (ADP), a variant of dynamic programming which uses approximation to evaluate control decisions, to solve these problem classes due to its effectiveness in practical applications [8]. Although relevant studies in railway related ADP research has been few and far in between [77, 82, 83] other problems tackled using ADP are similar to railway re-scheduling. Among the many streams of research into ADP, literature on traffic, scheduling and logistics problems are the most relevant to the present work.

Among them is the work by Cai et al. [9, 84], who investigate the application of ADP to road traffic signal control, aiming to develop a self-sufficient adaptive controller for online operation. In this work a parameterised linear approximation was employed for the approximation of the value function, that are updated progressively by adjusting the functional parameters in real-time. Two learning techniques were used in this work: temporal-difference (TD) learning and perturbation learning. The TD method updates the parameters by minimising the difference between current estimation and actual observation of state values. Whereas, perturbation learning sends a perturbing signal to the system state and estimates the gradients of the approximate function based on this perturbation. Despite of the different learning methods, the effects on performance were reported to be similar, and no statistical difference was found in their numerical experiments. The experiments from this work suggests that ADP with linear approximation worked well and therefore no more elaborate approximation methods were employed.

Similarly Yin et al. [85] used ADP to solve the traffic signal control problem, however as for the learning technique, the recursive least-squares temporal-difference (LSTD) learning was employed. Good performance of the ADP algorithms were reported in both of these works.

Another use of LSTD learning is by Davis et al. [86] who developed an ADP algorithm with linear approximation for solving a defensive, asset-based variant of the dynamic weapon-target assignment problem. In this work, it is reported that the ADP algorithm achieved an 8% to 22% mean optimality gap when compared to the globally optimal policy.

As for application of ADP in transportation in general, Powell et al. [87] report a unified framework for use of ADP in transportation and logistics. The stated aim of this work is to present ADP as a viable approach to stochastic optimisation problems arising in the fields of transportation and logistics.

Literature on ADP for scheduling problems is rich, with variety of ADP approaches being deployed to tackle these problems. Among the works, Papadaki and Powell [88] consider the problem of approximating optimal strategies for the batch service of customers at a service station with customers stochastically arriving at the station incurring a waiting cost and a service cost.

Schmid [89] uses ADP to solve a dynamic ambulance dispatching and relocation problem. Using observed data, it is reported that ADP provides better quality solutions compared to policies that are currently in use. The average response time of ambulances were reported to have been reduced by roughly 12%.

Li and Womer [90] investigated the use of ADP for project re-scheduling problems with resource constraints and stochastic task durations. In their approach an ADP algorithm is built upon integration of the look-back (lookup table) and look-ahead (approximation of the value function in DP) policies.

One of the earliest uses of TD learning for job-shop scheduling is that of Zhang and Dietterich [91] who developed a repair-based scheduler that starts with a critical-path schedule and incrementally repairs constraint violations.

Also, Ramírez-Hernández and Fernandez [92] present the application of an ADP algorithm to the problem of job releasing and sequencing of a benchmark re-entrant manufacturing line (RML). The approach is based on the SARSA(λ) algorithm with linear approximation architecture, whose parameters are updated through a gradient-descent approach. The

objective of this algorithm is to minimise inventory costs and maximise throughput. It is reported that a statistical match in performance between the optimal and the approximated policies obtained through ADP was observed through simulation.

Medury and Madanat [93] applied an ADP approach to planning and scheduling of the activities of an infrastructure management. They included making decisions on financial allocation of resources for activities, and the scheduling of the activities over a planning horizon of years. Given the complexity and the levels of uncertainty involved in this, a linear approximation architecture with TD learning was employed and they reported that ADP provided better results compared to commonly used approaches.

Van Roy et al. [94] presented an application of ADP to the optimisation of retailer inventory systems. The specific case under discussion in this work involves a model with thirty-three state variables, hence the encouragement to use ADP to solve this problem. A linear approximation function was employed with a variation of TD learning. The approach is reported to have outperformed well-accepted heuristics in retailer inventory management.

Other related ADP used to solve logistics problems include Katanyukul et al. [95] who investigated the application of ADP to an inventory problem. In this study a learning-based ADP method is compared with two simulation-based ADP methods: Rollout, and Hindsight Optimisation.

Using perturbation learning, Papadaki and Powell [96] developed an ADP approach for a batch dispatch problem in which times of arrivals are stochastic. They compared both linear and non-linear approximation functions, and reported the non-linear approximation to produce better solutions compared to linear approximation as iterations go on, however, it is noted that for large-scale problems they are very difficult to implement.

Schütz and Kolisch [97] considers the problem of capacity allocation with sequential delivery of service with stochastic service times. To tackle this problem a heuristic simulation-based ADP was employed based on the λ-SMART algorithm. Instead of solving the Bellman's equation for all states in each iteration, the λ-SMART algorithm proceeds along a simulated trajectory and only processes one state per iteration. Each time a state is visited

the optimal decision is made by solving the corresponding Bellman's equation based on the current estimate of the value function and the one-step cost. A comparison of the heuristic ADP and an optimal DP algorithm using deterministic service times shows the policies produce to be almost identical in average profit.

Papageorgiou et al. [98] introduces an ADP framework for generating solutions to a class of maritime inventory routing problem (MIRP) with a long planning horizon. It is reported that ADP is capable of producing better solutions compared to other well established approaches in MIRP that mostly use commercial MILP solvers. Their research uses a separable piecewise linear continuous function approximation for the value function.

2.4 Summary and discussion

Train movements are restricted to one degree of freedom and trains only follow the direction of rails. To eliminate the danger posed from colliding trains, railway control systems are employed to ensure safe operations for passengers and railway operators. It is not practical, especially in high-speed operations, to solely rely on reaction of train drivers to stop trains once an obstacle on the track, *e.g.* a slow moving train downstream, has been sighted, as the braking distance for trains often exceeds the sighting distance of train drivers. Therefore, control systems are put in place to inform train drivers of downstream traffic well in advance of required stopping points.

Railway traffic control is currently based on fixed block colour light signalling, where all tracks are divided into a series of fixed blocks. The principles of fixed block signalling are: to preclude concurrent presence of two trains in any block, and ensure the distance separating a train from the next one downstream is always greater than the full braking distance required for the rear train to stop safely. Under these principles trains are protected against colliding with one another and hence ensure safe operations. As the principles are enforced by controlling the colour of fixed signals, drivers can only react to instructions once a signal is sighted. The implication is, the under-utilisation of available capacity. This places

conflicting requirements upon the control system: those of safety and those of operational capacity.

In recent years, with advances in information and communications technology (ICT), railway undertakings have been investigating modern control systems to improve the conflicting requirements of the control system. Notable among these new systems is the European Train Control System (ETCS). Under ETCS-Level 2 and Level 3, by continuous wireless communication with the traffic control centre, a train does not wait until it approaches a signal or a balise (beacons that are placed strategically on the rail network and transmit data to trains as they pass over them) to get a new movement authority; it can receive them by radio at any time that sufficient blocks are cleared ahead of the train. The in-cab Driver Machine Interface will then display the movement authority beyond what may otherwise be observed from the signals, and when appropriate authorise operation at higher speeds. The aim of these new control systems is to take advantage of new technologies to improve capacity utilisation while at the same time ensuring safe operations at all time.

Envisioned railway control systems, such as ETCS, once in place and operational, present an opportunity in further improving capacity utilisation without the need of building new railway infrastructure. As complete information on all trains' real-time speeds and positions are transmitted to a control centre, approaches have been developed, in anticipation, to optimise railway traffic by considering current status of all trains and anticipating downstream traffic states. These approaches, referred to as real-time railway re-scheduling, have been widely studied in recent years, and conceptually demonstrated to be beneficial in terms of performance. Main benefits arise when deviations from the timetable cannot be recovered within the available time margins, and therefore a new schedule is computed that is more appropriate for the prevailing traffic state of the rail network.

Railway re-scheduling is a complex problem and as the size of the problem at hand increases so does the computational time required to solve for appropriate solutions. The quality of solutions and the size of the problem in re-scheduling have a direct impact on the computational time of the approaches presented in the literature, *i.e.* higher quality solutions

and larger problems increase the required computational effort. Furthermore, a full-scale evaluation and sensitivity study of re-scheduling approaches, with the possibility to draw quantitative and general conclusions about configuration and parameter setting is still missing, as the quantitative improvement of traffic performance due to these systems is difficult to evaluate.

One promising method in the field of optimisation and control that has attracted much attention in the recent decades is approximate dynamic programming (ADP) due to its effectiveness in tackling stochastic applications where the state and action spaces are large. This approach depends on an approximation to the value function of dynamic programming after optimisation from a specified state, and by using this approximation, the ADP avoids extensive explicit evaluation of performance and so reduces the computational burden substantially. The ADP is based on an algorithmic strategy that uses the result of each optimisation to update the current approximation of future performance. Through this updating, the ADP can adapt its optimisation in response to changes in the operating environment.

Given the robustness of ADP in tackling complex resource planning and control problems, as is presented in the literature, it is desirable to develop a re-scheduling framework based on ADP, to achieve real-time railway traffic control reliably and in a timely manner that meets the practical computational requirements of railway operations closely.

Chapter 3

Approximate Dynamic Programming

"Philosophy is written in that great book which ever lies before our eyes, I mean the universe. But we cannot understand it if we do not first learn the language and grasp the symbols, in which it is written." - Galileo Galilei

Even though dynamic programming (DP) provides exact solutions for optimisation over time, it suffers from the *curses of dimensionality*. This refers to the computational demand involved in calculations of DP which are exponential to the size of each of the state space, information space and decision space. A consequence of this is that the amount of computation required increases with time into the future whilst the relevance to immediate decisions decreases because of accumulated uncertainty.

In this chapter, we outline how approximate dynamic programming (ADP) reduces this computational burden while learning from online information constantly to adapt to the operating environment and so improve its decisions. Under the concept of reinforcement learning, specific techniques of learning are discussed, which form the basis of the adaptive ADP framework developed in this thesis.

3.1 Dynamic programming

DP deals with problems where decisions are made in stages, and the outcome of each decision can be anticipated to some extent before the next decision is made. One key aspect

of such problems is that decisions cannot be viewed in isolation and decisions made for the current state of the system may affect future states. DP is based on *Bellman's Principle of Optimality*, which states that:

> "An optimal policy [sequence of decisions] has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [11].

This asserts that an optimal policy has sub-policies that are optimal for their stages; therefore, replacement of a suboptimal decision with an optimal decision would improve the original sequence of decisions. To capture this, the DP technique evaluates decisions based on the sum of the present cost and the anticipated future cost, assuming optimal decision making for subsequent stages.

3.1.1 DP formulation and solution methods

Let $s \in S$ be a vector of state variables of the system, $u \in U$ the decision variable, and $g(.)$ the one step cost function. Given an initial state s_0, a sequence of decisions π_t made up of decisions $u \in U$ at step t, and future-discount factors α_t $(t \geq 0)$, a dynamic programme over horizon of T steps is

$$\min_{u_t \in U} E_v \left\{ \sum_{t=0}^{T} \alpha_t \, g(s_t, u_t, s_{t+1}) \mid s_0 \right\}. \tag{3.1}$$

A convenient form defined by

$$\alpha_t = e^{(-t\theta)}, \tag{3.2}$$

and θ is a discount rate for cost incurred in the future per step, s_{t+1} is the next state resulting from implementing u_t at step t

$$s_{t+1} = f(s_t, u_t, v_t),$$

$f(.)$ represents the state transition function, v_t represents random information at stage t, and E_v (E from this point on) is the expectation over this random information.

The optimal performance J at a single step t of the solution can be represented by solving *Bellman's* recursive equation [11]:

$$J(s_t) = \min_{u_t \in U} E\left\{g(s_t, u_t, s_{t+1}) + \alpha_{t+1} J(s_{t+1}(u_t))\right\}. \tag{3.3}$$

The backward dynamic programming procedure addresses this by starting from the final time T, assuming the optimal value function $J(s_T)$ is known, and solving equation (3.3) recursively working backward to the initial state s_0 to obtain the sequence of decisions which would lead to the optimal solution and the associated estimate of performance.

To guarantee that an optimal policy for the whole problem is found when state s_0 is reached using backward DP, the problem should possess the *Markovian property*. Given that P is a matrix of transition probabilities, a process is said to have the Markovian property if

$$P\{s_{t+1} = x \mid s_0,\ s_1,\ \ldots,\ s_t\} \ = \ P\{s_{t+1} = x \mid s_t\},$$

for $t = 0,\ 1,\ \ldots$, and every sequence $s,\ x,\ s_0,\ s_1,\ \ldots,\ s_{t-1}$. This implies that knowledge of the current state of the system conveys all the information about its previous behaviour necessary for determining the optimal policy henceforth.

Bellman's equation can be used to describe deterministic and stochastic problems. A deterministic DP requires the transition function to the next state s_{t+1} of the problem to be completely determined by the current state s_t and the current decision u_t. In such a case, the Bellman equation for the problem is described as

$$J(s_t) = \min_{u_t \in U} \left\{g(s_t, u_t, s_{t+1}) + \alpha_{t+1} J(s_{t+1}(u_t))\right\}, \tag{3.4}$$

whereas the transition function to the next state s_{t+1} of a stochastic problem, cannot be fully determined by the current state s_t and the current decision u_t. Such a problem can be

formulated as a *Markov Decision Process*, where the Bellman equation can be reorganised for a stochastic problem as

$$J(s_t) = \min_{u_t \in U} \left\{ \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, u_t) \Big[g(s_t, u_t, s_{t+1}) + \alpha_{t+1} J(s_{t+1}(u_t)) \Big] \right\}, \tag{3.5}$$

where $p(s_{t+1}|s_t, u_t)$ is the probability of getting to state s_{t+1} by taking decision u_t in state s_t.

DP can formulate problems with finite and infinite horizons. A problem formulated in DP is said to have a finite horizon if the value function $J(.)$ accumulates over a finite number of steps; whereas a problem is said to have an infinite horizon if value function $J(.)$ accumulates indefinitely. This can be expressed in equation (3.1) by the choice of T.

By considering railway traffic as a finite horizon problem it is possible to define the final value function $J(s_T)$ as the overall performance of the finite steps; on the other hand defining the final value function for an infinite horizon operation is not easily done. Fortunately, the infinite horizon problem leads to steady-state in Markov process. At steady-state, the state transition probabilities become time-invariant, and the value of $J(.)$ converges. This allows for derivation of analytical solutions to our control problem which can be tackled by iteration algorithms. There are two common iteration algorithms: *policy iteration* and *value iteration*.

3.1.1.1 Policy iteration

Policy iteration terminates in finite iterations even for infinite horizon problems. This iteration algorithm essentially finds the value of specific policies, by starting from an initial policy π_0 (a sequence of decisions u) and generating a sequence of new policies π_1, π_2, π_3, ..., π_m. Policy iteration has two steps; i) policy evaluation, and ii) policy improvement. Let s' represent the next state, policy evaluation is

$$J^{\pi}(s) = \sum_{s' \in S} p(s'|s, u) \Big[g(s, u, s') + \alpha J(s'(u)) \Big], \tag{3.6}$$

Input: P, α
Set $m = 0$ and select $\pi^m \leftarrow random(U)$,
while *Policy is not stable:* **do**
  for *each* $s \in S$*:* **do**
    Evaluate policy using (3.6),
  end
  for *each* $s \in S$*:* **do**
    Improve policy using:
    $\pi^{m+1}(s) = \underset{u \in U}{\arg\min} \left\{ \sum_{s' \in S} p(s'|s,u) \left[g(s,u,s') + \alpha J(s'(u)) \right] \right\}$,
  end
  if $\pi^m = \pi^{m+1}$ *:* **then**
    Policy is stable: Stop.
  else
    $m \leftarrow m+1$,
    $\pi^m \leftarrow \pi^{m+1}$,
  end
end

Algorithm 1: Policy iteration.

and for policy improvement we find the decision that minimises equation (3.6). Algorithm 1 presents the policy iteration procedure. The proof of the convergence of policy iteration can be found in Section 2.2.3 of [99].

3.1.1.2 Value iteration

Value iteration is perhaps the simplest algorithm in DP and is widely used to solve many different problems in a wide range of topics; it is also the principal method for calculating the optimal value function J^*. Generally, the method requires an infinite number of iterations to converge to J^*, however, in some circumstances the method can terminate finitely. Because value iteration requires an infinite number of iterations to obtain the exact value function J^*, a termination criterion is required in this algorithm. Algorithm 2 summarises value iteration procedure.

Value iteration is similar to the backward dynamic programming algorithm; instead of starting from the last stage T and working backward to 0, an iteration counter is used that starts at 0 and increases until the convergence criterion is satisfied. The proof of convergence for value iteration can be found in Section 3.10.2 of [12].

Input: P, α
Set $J^0(s) = 0 \; \forall s \in S$, set termination criterion κ, set $m = 1$,
while $\Delta > \kappa$*:* **do**
  $\Delta \leftarrow 0$,
  for *each* $s \in S$*:* **do**
    $J^m(s) \leftarrow \min_{u \in U} \sum_{s' \in S} p(s'|s,u)\Big[g(s,u,s') + \alpha' J(s'(u))\Big]$,
    $\Delta = \max\,(\Delta, \parallel J^m(s) - J^{m-1}(s) \parallel)$
    if $\Delta < \kappa$*:* **then**
      Stop.
    else
      Continue,
    end
  end
  $m \leftarrow m+1$
end
Output policy, such that:

$$\pi(s) = \arg\min_{u \in U} \left\{ \sum_{s' \in S} p(s'|s,u)\Big[g(s,u,s') + \alpha J(s'(u))\Big] \right\}.$$

Algorithm 2: Value iteration algorithm for infinite horizon optimisation.

3.1.2 Curses of dimensionality

Although equation (3.3) characterises an elegant way of representing an optimisation problem, it can be computationally expensive and therefore is not always practical for operational use. The reason for this is the need to consider the entire state space to evaluate the optimal decision at each step, and the computational intensity grows exponentially with additional states in the space. This intensity in computational demand is caused by the *three curses of dimensionality* [12]:

1. State space: If the state variable $s_t = (s_{t_1}, s_{t_2}, \ldots, s_{t_I})$ has I dimensions, and if s_t can take on S possible values, then there may exist up to S^I different states.

2. Decision space: The decision vector $u_t = (u_{t_1}, u_{t_2}, \ldots, u_{t_K})$ might have K dimensions. If u_t can take on U decisions, then there may exist up to U^K decisions.

3. Outcome space: The random information variable $v_t = (v_{t_1}, v_{t_2}, \ldots, v_{t_W})$ might have W dimensions. If v_t can take on O outcomes, then there may exist up to O^W outcomes.

Recursively solving the Bellman equation requires that at each step t, for each combination of state, decision and outcome, a value function J be calculated to make an optimal decision. Therefore, the computational demand would increase exponentially with as many as:

$$S^I \times U^K \times O^W.$$

This requirement means to calculate all value functions, we have to loop over three nested loops with the first one looping over all states, and the last loop looping over all outcomes. This clearly is not practical where state and decision spaces are large and stochastic, and cannot be solved efficiently using the direct approaches presented in section 3.1.1.

Deployment of DP for real-time control is especially onerous. For one, considering the entire state space to the end of the planning period at every step may not be efficient, as plans for future decisions might need to be adjusted in light of emergence of new information, and decisions planned in the 'tail' period may never be implemented.

Furthermore, complete information on future states may not be available owing to the stochastic nature of the control environment. Although the opportunity will often arise to revise these decisions as their time approaches, calculating them in the first place generates a heavy computation burden.

To adopt DP for real-time control applications, one must look to reduce the dimensionality of the control problem and look to approximate future performance where possible. In the next section we discuss how this may be done by trading off solution quality for lower memory and computation requirements.

3.2 Fundamentals of ADP

Section 3.1.1 presented how to solve optimisation problems of the form of equation (3.1)

$$\min_{\pi_t \in U} E\left\{ \sum_{t=0}^{T} \alpha_t \; g(s_t, u_t, s_{t+1}) \mid s_0 \right\},$$

by recursively calculating equation (3.3)

$$J(s_t) = \min_{u_t \in U} E\left\{g(s_t, u_t, s_{t+1}) + \alpha_{t+1} J(s_{t+1}(u_t))\right\}.$$

From this point on we will use equations (3.1) and (3.3) to formulate our solution methods, which encompass all requirements necessary to formulate our railway traffic management problem.

Based on the presentation in section 3.1.2, we showed that equation (3.1) can be computationally expensive and intractable even for small problems. Furthermore, it is evident that for many applications where the state and decision spaces are large, even the optimality equation (3.3) can itself be computationally expensive and intractable.

To address the difficulties associated with DP, the approximate dynamic programme (ADP) approach to decision-making has been developed. ADP works forward in time and employs a greedy approach to make decisions using available information in an exhaustive search for decisions in the short-term future and an approximation to the value function $J(.)$ after that.

This approach, as is set out in the remainder of this section, is intended to tackle problems that have large state, decision and information spaces, but also to tackle problems where a model of the system does not exist or is too complex to formulate effectively. In the remainder of this chapter we will look at solutions to all of these problems, though the main focus of this investigation is to develop ADP for railway traffic management for which a model of the system can be reasonably formulated and as such ADP for *'large problems'* is the main focus of this thesis.

ADP is an exciting and still emerging method to solve difficult optimisation problems, and it offers a large range of solution approaches to choose from. In this investigation we focus on model-based approaches which use reinforcement learning as part of the solution, there are however other similar approaches based on ADP. Which of the approaches is preferred is a question that can only be answered by applying a range of different approaches to a

problem and comparing the results. For instance, Cai et al. [9], tried two different ADP approaches and reported similar performance and no statistically significant difference in performance in a traffic signal control application.

In the remainder of this section we look at the fundamentals of ADP and the idea underlying this approach to show how ADP can tackle the *three curses of dimensionality* in DP.

3.2.1 Making decisions approximately

The most computationally inefficient part of backward DP is to compute the value function $J(.)$ in equation (3.3) for all steps from the final stage backward to the current stage. This procedure is therefore the prime candidate to be eliminated to improve computational efficiency of DP. However, there still is a need to somehow measure the long term performance, so to achieve this $J(.)$ can be *approximated*. Let $\widehat{J}(s)$ be an approximation of the optimal value function, then

$$J(s) \approx \widehat{J}(s),$$

where $\widehat{J}(s)$ is an approximation of long term performance resulting from visiting state s. Adopting this in Bellman's equation results in

$$\widetilde{J}(s_t) = \min_{u_t \in U} E\left\{ g(s_t, u_t, s_{t+1}) + \alpha_{t+1} \widehat{J}(s_{t+1}(u_t)) \right\}, \tag{3.7}$$

Where we refer to $\widetilde{J}(s_t)$ as an observation of value function at state s_t. Making decisions using this framework is then to solve

$$u_t = \arg\min_{u_t \in U} E\left\{ g(s_t, u_t, s_{t+1}) + \alpha_{t+1} \widehat{J}(s_{t+1}(u_t)) \right\}. \tag{3.8}$$

which is referred to as a *greedy strategy* because of the implicit exhaustive search over decisions ate stage t.

This is the first step for the transition from DP to ADP, and is where the *approximate* in ADP comes from. This is a structural change to how problems are viewed. Instead of having to compute what the exact value function is, which in the case of backward DP requires calculation of all J from the final stage to the current stage, we adopt the approximation $\widehat{J}$ for the value function.

This of course introduces errors in calculations, which needs to be addressed. The challenge of ADP is to produce an effective approximation, which we apply by iterative improvement. Because approximations only need to be as good as is required to solve the problem at hand "sufficiently", however it is difficult to decide on ad-hoc estimations of $\widehat{J}$ that results in sufficient system performance. It is possible to assume or compute, then store in a table all approximate value functions $\widehat{J}$ that result from taking decision u in state s, but this can't be efficient when the state space is large and in many cases may not be sufficiently close to the optimal value function J to result in appropriate decisions. Therefore, ADP adopts a systematic approach to refine the approximate value function $\widehat{J}$, which is presented in section 3.2.4.

3.2.2 Stepping forward in time

For now assuming a well defined approximate value function $\widehat{J}$ is available for the current state s_t, it is possible to find the decision u_t that minimises the Bellman's equation (3.3), which is given by equation (3.8). Being able to make a decision for the current state whilst accounting for all possible future decisions, eliminates the need to consider all stages to find the value function, and therefore stepping back in time can be avoided. Hence, ADP steps forward in time to move into the next stage s_{t+1}, receive information from the system and repeats this process progressing forward in time.

In cases where the environment is stochastic, the expected value of near-future decision cost $E_v(g(s,u,s'(s,u)))$ (the one-step cost from this point on) in equation (3.3) may be evaluated using a *Monte Carlo scheme*.

Monte Carlo scheme or Monte Carlo simulation is a sampling approach which samples from

the distribution of the random value to estimate the expected outcome of a process, where larger sample sizes produce better evaluations. Therefore, this process generates a sample path for the system. As for the approximate value function $\widehat{J}$, in this thesis we assume that the transitions are determined by $s,\ u$ and v. In the case of probabilistic transitions, $\widehat{J}$ should be evaluated by taking into account the one-step transition according to

$$\sum_{s_{t+1}\in S} p(s_{t+1}|s_t,u_t)\left[\alpha_{t+1}\widehat{J}\left(s_{t+1}(u_t)\right)\right]. \tag{3.9}$$

By stepping forward in time, all the steps required to compute the value function J in equation (3.3), which informs us of the long term performance based on the current state, can be avoided.

This is precisely how ADP improves computational efficiency significantly compared to DP. Instead of visiting all states, ADP focuses on the near-future for which information is more reliable and decisions imminent, and avoids calculations that require unnecessarily fine information and lie beyond the near-future.

Because ADP works forward in time, it requires an approximate value function $\widehat{J}$ for states which may be encountered in future, some of which might not have been visited before. As was discussed in the previous section, a table representation of all $\widehat{J}$ may not be sufficient if a new state is visited. This brings us to *functional approximation* which is discussed in the next section.

3.2.3 Linear function approximation

A common approach to approximate the value function and coping with large state spaces is to use a separable linear approximation of the form

$$\widehat{J}\left(s,\ r\right) = r^{\mathsf{T}}.\ \phi(s), \tag{3.10}$$

where $r \in \mathbb{R}^M$ is a vector of parameters of the approximation function, and $\phi : S \to \mathbb{R}^M$ is a features extraction function (or basis function) defined on the state space S that maps the

state to a feature vector for which r is the associated weight vector and M is the number of features used. The purpose is to represent the contribution of each feature to the value function, and hence, approximate the value function by inspecting the state.

Some practitioners, avoid approximating the value function $\widehat{J}$ using the same features for different decisions [14]. Therefore it is common to distinguish between value functions resulting from different decisions from the same state by defining slots in the feature vector for each decision and populating the slots in the vector by 0 if the decision is not going to be evaluated: assuming the decision space U has only two possible decisions that can be chosen, i.e. u_1 and u_2, and only two features are extracted from state s, i.e. ϕ_1 and ϕ_2, then the features are defined as

$$\phi(s) = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}, \ \phi(s, u_1) = \begin{bmatrix} \phi_1 \\ \phi_2 \\ 0 \\ 0 \end{bmatrix}, \ \phi(s, u_2) = \begin{bmatrix} 0 \\ 0 \\ \phi_1 \\ \phi_2 \end{bmatrix}, \tag{3.11}$$

which distinguishes between the two possible decisions. This is especially useful when the decisions result in the same short term performance but may result in different long-term performance. In this investigation, in cases where features do not distinguish between decisions, equation (3.11) is used to represent feature vectors; otherwise, the said representation is not used.

For problems that are too complex to be solved analytically, functional approximations can be used to approximate the value function by representing it in a lower dimensional space using $M \ll |S|$ parameters. A particular convenient form of this is linear:

$$\mathbf{J}(r) \approx \widehat{\mathbf{J}}(r) = \begin{bmatrix} -\ \phi^{\mathsf{T}}(s_1)\ - \\ -\ \phi^{\mathsf{T}}(s_2)\ - \\ \vdots \\ -\ \phi^{\mathsf{T}}(s_{|S|})\ - \end{bmatrix} \times \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_M \end{bmatrix} \equiv \Phi_{|S|\times M}\, \mathbb{R}_{M\times 1}. \tag{3.12}$$

By now the computational efficiency of ADP to solve intractable DP problems has become evident and now we turn to methods of improving the approximations by updating r, which gives ADP its adaptive character.

3.2.4 Learning from experience

The use of features ϕ from states in the approximate the value function means that changes in the states can be tracked based on decisions taken at each state. An opportunity to improve an approximation arises after each optimisation, when it is possible to compare the approximate value function and the observed value function.

Given that a functional approximation is employed, then it is possible to update the vector of functional parameters r by finding the solution to

$$r^* = \arg \min_{r \in \mathbb{R}^M} \| J^* - \widehat{J}(r) \|, \tag{3.13}$$

where J^* is the newly optimised value function. Therefore, estimation of the value function is achieved by using equation (3.13) recursively to update the parameter vector r. This is termed 'learning' in ADP community and lies at the heart of any ADP approach. Indeed, Tsitsiklis and Van Roy [15] proved that for linear function approximation this approximation process for the parameters converges to the unique optimal parameter vector r^*.

As more states arise and are optimised, experience can be accumulated to improve approximations over time. In the present case of railway traffic management in particular, and traffic management in general, learning can therefore adapt the parameters r according to prevailing traffic conditions on the network and calculate decisions accordingly. As we will see by the end of this chapter, this adds an adaptivity to the controller so that it can be regulated to react to changes in traffic with varying degrees of sensitivity.

There are several different methods for updating the parameter vector r, many of which use *Machine Learning* techniques to calculate adjustments Δr to r that would minimise equation (3.13), which are usually moderated using

$$r_{t+1} = r_t + \eta_t \Delta r_t, \tag{3.14}$$

where $\eta_t \in (0,\ 1]$ $(t \geq 1)$ is a decreasing sequence of moderating factors that satisfy the following conditions for convergence:

$$\sum_{t=0}^{\infty} \eta_t = \infty,\ \sum_{t=0}^{\infty} {\eta_t}^2 < \infty. \tag{3.15}$$

Among the machine learning techniques, *Reinforcement Learning* techniques have received considerable attention in the literature and have been adopted successfully to ADP approaches of adaptive traffic management.

Temporal-Difference Learning introduced by Sutton [100], a reinforcement learning technique, is of particular interest in this thesis as it forms the basis of the learning mechanisms developed for the application of railway traffic management. Reinforcement learning and temporal-difference learning are presented in sections 3.3 and 3.4 respectively.

3.2.5 An ADP algorithm

All the steps presented in sections 3.2.1 to 3.2.4, describe one single decision and transition forward to the next state. Running through these steps iteratively is intended to yield improvement in terms of decision-making. This is implemented by solving equation (3.13) and adopting adjustments to r using equation (3.14) to adjust the value function approximation with the intention of improving it and so to make better decisions in the future.

An ADP algorithm is presented in Algorithm 3 [12]. This differs from backward DP in that it progress forward in time considering all possible decisions and the states that result from them rather than evaluating all states by working backward from the final stage.

Using ADP allows for achieving sequential decision-making in real-time, and given the reduced computational effort, there is opportunity for revisiting decisions already planned if the environment changes. Furthermore, learning from experience plays a balancing act between the immediate performance and that of the long-term.

Step 1. Initialisation:

Initialise parameter vector r_0, initiate learning rate η_0 (where required);

Set $t = 0$, choose the horizon T stages to be calculated explicitly;

Step 2. Receive current state s:

Set $t = t + 1$;

Step 3. Produce decisions and update $\widehat{J}$:

for *each* $t \in T$*:* **do**

Find the optimal decision u_t^* using

$$u_t^* = \arg\min_{u_t \in U} E\left\{g(s_t, u_t, s_{t+1}) + \alpha_t \widehat{J}\left(s_{t+1}(s_t, u_t), r_t\right)\right\},$$

Calculate new observation $\widetilde{J}\left(s_t\right)$ using

$$\widetilde{J}\left(s_t\right) = E\left\{g(s_t, u_t^*, s_{t+1}) + \alpha_t \widehat{J}\left(s_{t+1}(s_t, u_t), r_t\right)\right\},$$

Update approximation based on experience by

$$r_{t+1} = r_t + \eta_t \left[\arg\min_{r \in \mathbb{R}^M} \| \widetilde{J}\left(s_t\right) - \widehat{J}\left(s_t, r\right) \| - r_t\right],$$

end

Step 4. Implement decision:

Implement decisions π_t^*, *ie.* $[u_t^*, \cdots, u_T^*]$, and transition to s_T;

Return to step 2.

Algorithm 3: An ADP algorithm.

3.3 Reinforcement learning

In many practical applications the behaviour of the environment may not be modelled exactly due to the complexity of the system. In such cases, *Reinforcement Learning* (RL)

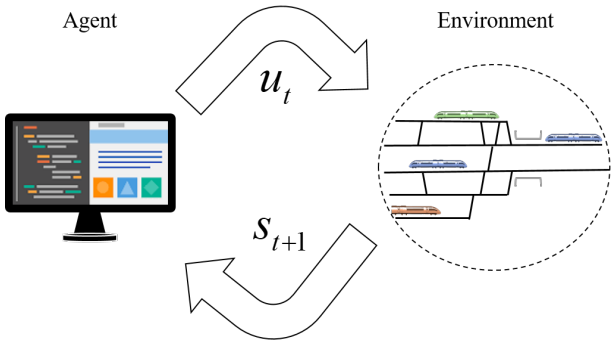


Figure 3.1: The agent-environment interaction in reinforcement learning in the case of railway traffic management.

can be employed to mimic an intelligent beings approach to learning the dynamics of the environment. Learning from interaction is the foundation of RL, where an *agent* (a system that is learning from a control process) interacts with the environment on each step, using an available decision and then observes the cost incurred from that decision. The idea is to discover which decisions yield the least cost by trying many decisions randomly and observing their outcomes.

In the case of railway traffic control, RL would choose scheduling decision u_t and observe cost incurred at the next state s_{t+1} from u_t taken at state s_t; *ie.* $g(s_t, u_t)$. This is shown in Figure 3.1.

RL procedure requires the agent to estimate the relationship between costs and states by observing the environment and incrementally updating its estimation of outcomes. To do so RL pairs decisions u_t and states s_t and stores an expected cost for this pair, *ie.* $\widehat{J}(.)$. The observed cost is then used to incrementally update the cost associated with the decision-state pairs, which is a similar procedure as (3.13).

It is clear that RL involves significant abstraction of any problem, and it is only applicable to goal-driven problems that can interact with the environment in someway, *eg.* actual interactions with the environment until the agent learns the optimal policy or simulation based interaction which can be setup to represent a practical environment. Therefore, RL states that any control problem can be simplified to three fundamental *signals*:

- Decisions: The choices made by the agent (controller);
- States: The basis on which the choices are made; and
- Values (reward or cost): The objective by which performance is measured.

Not all control problems can be represented using the above signals, though many problems can be, including the present railway traffic management problem. How decisions and states are represented can influence RL performance, and studies into such representational choices remain an active area of RL research [14].

In the present investigation the way in which decisions and states are represented is decided explicitly rather than by an automatic representation procedure. The focus of this investigation is on application of ADP and RL principles for learning and how to make sequential decisions rather than effects of different decisions and states representation.

The value signal allows for formal representation of the goal of the problem. RL agent minimises the total amount of cost it receives from the environment. This means minimising the long term cost is the objective, rather than just the immediate cost. All RL techniques consider representing the long term cost as best as is possible in sufficient computational time.

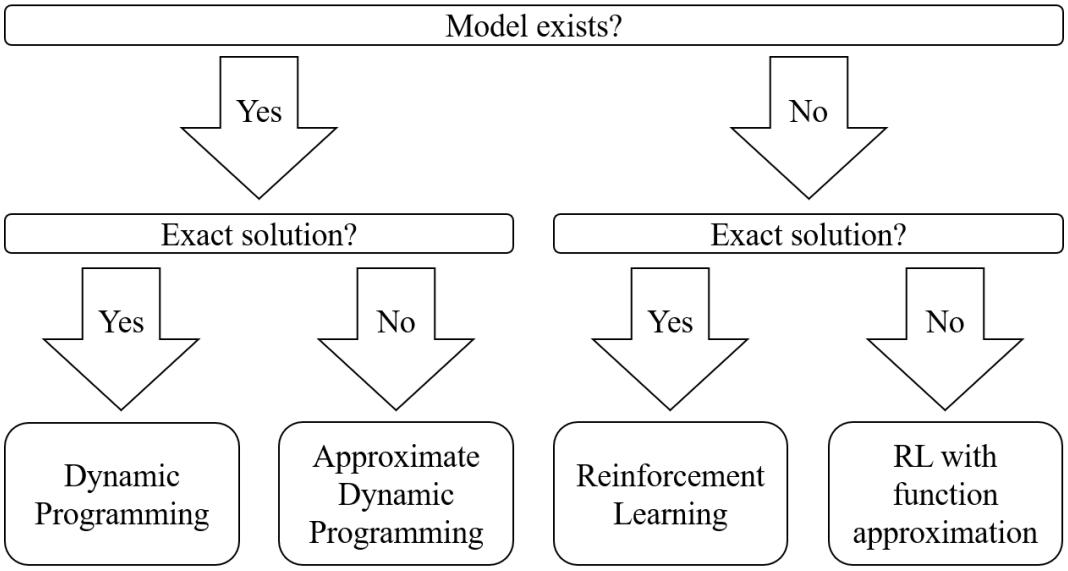


Figure 3.2: Bellman equation solvers separated based on availability of a model of the system, and use of approximation.

Therefore, the cost itself should represent the goal of the agent rather than knowledge on *how* to achieve the goal. It is the aim of RL to allow for the agent to explore different possibilities to achieve the goal and the agent should not be restricted in how to achieve this goal, unless practical realities dictate constraints which should be represented. For example, the goal of an agent in learning how to play chess is to win a match rather than achieving sub-goals of taking the most opponent's pieces out of play. If sub-goals are costed then RL might focus on achieving the sub-goals as oppose to winning the match.

RL relies on the Markov property, described in the preceding sections, to represent decisions and costs as functions of the current state. This is important in RL as it implies that the best policy for choosing decisions as a function of a state is similar to choosing decisions as a function of complete history of all state transitions. If an environment has the Markov property, then using the one-step dynamics of the system allows for calculation of the next state and the expected cost associated with this transition. By using this principle iteratively, it is possible to estimate future expected costs from the knowledge of the current state which is equivalent to using the complete history up to the current time. It is clear that Markov states provide the best possible basis for choosing decisions and so RL utilises the Markov property to the full and, similar to ADP, RL tasks that satisfy this property are called Markov decision processes (MDPs).

It is evident that RL shares many fundamental ideas with ADP. Figure 3.2 distinguishes between different methods presented in this thesis to solve the Bellman equation. The remainder of this chapter describes RL methods that appear in the literature and shows how they can be viewed as ADP techniques.

3.3.1 Q-learning and SARSA

Similar to ADP, almost all RL algorithms are based on an estimation of the value function - which we denote as J - to inform of the outcome of choosing an decision in a particular state, where outcome here is defined in terms of the expected long term cost. The long term costs are dependent on a series of decisions and therefore the value function is defined with

respect to a policy. Recall that a policy π, is a mapping from each state $s \in S$, and decision $u \in U$, to the probability $p(s,u)$ of taking decision u when in state s. Therefore, the value of state s under a policy π, is the expected cost of starting in state s and following π thereafter to the end of horizon T:

$$J^{\pi}(s_t) = E\left\{\sum_{i=t}^{T} \alpha_{t-i}\, g(s_{t+i}, u_{t+i})\right\}, \tag{3.16}$$

where g, as before, is the one-step cost.

Similarly, the value of taking decision u in state s under a policy π can be defined as:

$$Q^{\pi}(s_t, u_t) = E\left\{\sum_{i=t}^{T} \alpha_{t-i}\, g(s_{t+i}, u_{t+i})\right\}, \tag{3.17}$$

where Q^{π} is the decision-value function for policy π. This function in RL community is referred to as the *Q-function*. The aim of RL is to minimise the *Q-function* over the long-term horizon.

Q-Learning is one of the oldest RL algorithms which was introduced in 1992 [101], and is named after the variable $Q(s,u)$. Given the definition of a Q-function in equation (3.17), Q-Learning chooses decision u at time t using

$$u_t = \arg\min_{u_t \in U} Q(s_t, u). \tag{3.18}$$

Note that to find u_t according to equation (3.18) and using an approximated Q-function, it is not necessary to compute an expectation or calculate the Q-value of being in a downstream state. This is a similar greedy approach to ADP's equation (3.8)

$$u_t = \arg\min_{u_t \in U} E\left\{g(s_t, u_t, s_{t+1}) + \alpha_{t+1}\widehat{J}\left(s_{t+1}(u_t)\right)\right\},$$

however the difference here is that Q-learning does not compute the expected one-step cost g. This eliminates the necessity of providing a model to the system and hence, Q-learning can be used to make decisions for complex systems that can not easily be modelled.

Once a decision u is implemented, a one-step cost g can be observed, and used to calculate an updated value of being in state s_t and taking decision u_t using

$$\widetilde{Q}\left(s_t, u_t\right) = g(s_t, u_t, s_{t+1}) + \alpha_{t+1} \min_{u_{t+1} \in U} Q(s_{t+1}, u), \tag{3.19}$$

where $\widetilde{Q}\left(s_t, u_t\right)$ is an observation of the Q-function based on a single interaction and can be used to update the Q-value by

$$Q^{+}(s_t, u_t) \leftarrow (1-\gamma_t) Q^{-}(s_t, u_t) \; + \; \gamma_t \widetilde{Q}\left(s_t, u_t\right), \tag{3.20}$$

where Q^{+} and Q^{-} are the updated and current Q-functions respectively, and γ is a step-size between 0 and 1.

Similar to ADP, if there exist a large state-decision space then the Q-function can be represented as a parameterised function, as in equation (3.10):

$$Q(s, u, r) = r^{\mathsf{T}} . \, \phi(s, u), \tag{3.21}$$

and one can adjust the parameters r to better match the observed costs iteratively over time. The resulting algorithm is called Q-learning and is shown in Algorithm 4 [101].

The δ in Algorithm 4 [101], is the error between an estimate of the Q-function based on a single interaction, $\widetilde{Q}$, and the current estimate of the Q-function. This error is referred to as the *temporal-difference* (TD) error, which we will explore in more detail in the next section and will form the basis of the learning mechanism adopted here for ADP.

Algorithm 4 may diverge when function approximation is employed [13], as is the case in our representation of the framework. This stems from the fact that the policy used for sampling, is not the same as the policy used for learning. However, matching these two

policies will lead to a convergent on-line method called *SARSA* introduced by Rummery and Niranjan [102]; this name refers to state-action-reward-state-action sequence which is the procedure for SARSA and is shown in Algorithm 5 [102].

Initialisation:

Initialise parameter vector r_0 and learning rate η_0;

Receive current state s_0 and policy π.

while *time left* **do**

Send $u_t \leftarrow \pi(s_t)$,

Receive cost g and next state s_{t+1},

$\widetilde{Q}(s_t, u_t, r_t) = g(s_t, u_t, s_{t+1}) + \alpha_{t+1} \min_{u_{t+1} \in U} Q(s_{t+1}, u, r_t)$,

$\delta_t = \widetilde{Q}(s_t, u_t, r_t) - Q(s_t, u_t, r_t)$,

$r_{t+1} = r_t + \eta_t \delta_t \phi(s, u)$,

$t = t + 1$.

end

Algorithm 4: Q-Learning.

Initialisation:

Initialise parameter vector r_0 and learning rate η_0;

Receive current state s_0 and policy π.

while *time left* **do**

Send $u_t \leftarrow \pi(s_t)$,

Receive cost g and next state s_{t+1},

$u_{t+1} \leftarrow \pi(s_{t+1})$

$\widetilde{Q}(s_t, u_t, r_t) = g(s_t, u_t, s_{t+1}) + \alpha_{t+1} Q(s_{t+1}, u_{t+1}, r_t)$,

$\delta_t = \widetilde{Q}(s_t, u_t, r_t) - Q(s_t, u_t, r_t)$,

$r_{t+1} = r_t + \eta_t \delta_t \phi(s, u)$,

$t = t + 1$.

end

Algorithm 5: SARSA.

In SARSA, $\widetilde{Q}$ is calculated based on $Q(s_{t+1}, \pi(s_{t+1}), r_t)$, rather than $\min_{u_{t+1} \in U} Q(s_{t+1}, u, r_t)$, which makes both the sampling and learning policies identical. Therefore, the agent learns from the same policy used for generating the trajectory. This is termed *on-policy* learning as opposed to *off-policy* learning in Q-Learning.

3.3.2 Actor-critic methods

In RL implementation, Q-learning and SARSA can be directed to visit all state-decision pairs infinitely often, by using an ε-greedy policy that choices plans greedily most of the time and at other times (with small probability ε) choices randomly from amongst all the actions available, whereas, actor-critic (AC) algorithms are based on the simultaneous on-line estimation of the parameters of two structures, called the *actor* and the *critic*. In this structure the policy-selection is represented as a separate entity, referred to as the *actor* and is implemented as the Gibbs softmax distribution [13] which maps states to decisions in a probabilistic manner:

$$\pi^{actor}(s,u) = \frac{exp(\frac{\rho(s,u)}{\tau})}{\sum_a exp(\frac{\rho(s,a)}{\tau})}\ , \tag{3.22}$$

where $\rho(s,u)$ is the preference of taking decision u in state s, and $\tau \in \mathbb{R}^+$ can be used to shift between greedy with respect to Q values (large τ) and random decision selection ($\tau = 0$). Therefore, $\pi^{actor}(s)$ returns the decision u with probability $\pi^{actor}(s,u)$.

The *critic* on the other hand is a conventional RL update rule for Q, that "criticises" the agent for the observed TD error. Therefore, the critic is concerned with the prediction problem, whereas the actor addresses the control problem. The AC algorithm solves these problems simultaneously to find an appropriate policy, as in policy iteration.

In applications where large state-decision spaces exist, $\rho(s,u)$ can be represented using a linear function approximation as

$$\rho(s,u,\omega) = \omega^{\mathsf{T}}.\ \psi(s,u)\ , \tag{3.23}$$

where $\omega \in \mathbb{R}^M$ is a column vector with each entry a parameter of the approximation function similar to r, and $\psi : S \rightarrow \mathbb{R}^M$ is a features extraction function similar to ϕ, defined on the state space S and so maps the state to a feature vector for which ω is the associated weight vector and M is the number of features used. It is noted that ψ and ϕ can be different and $\psi(s,u)$ is constructed similar to $\phi(s,u)$ presented in equation (3.11).

There are various ways of updating the actor [103]. Algorithm 6 [103] presents this framework based on the template introduced by Bhatnagar et al. [103].

Initialisation:

Initialise parameter vectors r_0 and ω_0, and learning rate η_0;

Receive current state s_0 and decision $u_0 \leftarrow \pi^{actor}(s_0)$.

while *time left* **do**

Send $u_t \leftarrow \pi^{actor}(s_t)$,

Receive cost g and next state s_{t+1},

$u_{t+1} \leftarrow \pi^{actor}(s_{t+1})$,

$\widetilde{Q}\ (s_t,u_t,r_t) = g(s_t,u_t,s_{t+1}) + \alpha_{t+1}Q(s_{t+1},u_{t+1},r_t)$,

$\delta_t = \widetilde{Q}\ (s_t,u_t,r_t) - Q(s_t,u_t,r_t)$,

Critic update used for $Q(r)$: $r_{t+1} = r_t + \eta_t\delta_t\phi(s,u)$,

Actor update used for $\psi(\omega)$: $\omega_{t+1} = \omega_t + \eta_t\delta_t\psi(s,u)$,

$t = t+1$.

end

Algorithm 6: Actor-Critic.

3.4 Temporal-difference learning

Temporal-difference (TD) learning is a widely used form of reinforcement learning [104]. TD methods can be used to learn directly from experience without a model of the environ-

ment's dynamic, and updates to the estimates are made without needing a final outcome.
A simple method to update the long term estimate of cost $\widehat{J}$ is to observe the accumulated actual cost of all decisions to the end of horizon T, and update the estimation based on this experience by

$$\widehat{J}(s_t) \leftarrow \widehat{J}(s_t) + \gamma_t \left[\sum_{i=t}^{T} \alpha_t \; g(s_i, u_i, s_{i+1}) - \widehat{J}(s_t) \right]. \tag{3.24}$$

However, TD learning, first introduced by Sutton [100], uses experience more immediately to improve future estimates. TD compares current estimates of the long term cost $\widehat{J}(s_t, r_t)$ with an observation of the one-step cost of transition $g(s_t, u_t, s_{t+1})$ plus a discounted future value $\widehat{J}(s_{t+1}, r_t)$ in the environment to produce an error in estimation called a *temporal-difference* error δ. This is calculated as

$$\delta_t = g(s_t, u_t, s_{t+1}) + \alpha_t \widehat{J}(s_{t+1}, r_t) - \widehat{J}(s_t, r_t), \tag{3.25}$$

where δ is used to update the current estimate of $\widehat{J}$ after every step. In the simplest form of TD, $\widehat{J}$ is updated by

$$\widehat{J}(s_t) \leftarrow \widehat{J}(s_t) + \gamma_t \delta_t \; . \tag{3.26}$$

Hence, this form of TD updates approximations as soon as information becomes available which is a more immediate form of learning compared to equation (3.24).
In the case of linear function approximation of equation (3.10)

$$\widehat{J}(s, \; r) = r^{\mathsf{T}} . \; \phi(s),$$

TD learning can be used to update the vector of the functional parameters r . We define a projection operator Π as

$$\Pi J \; = \; \arg \min_{\widehat{J} \in \left\{ r^{\mathsf{T}} . \; \Phi | r \in \mathbb{R}^M \right\}} \| \, J - \widehat{J} \, \| \; , \tag{3.27}$$

where J is the optimal performance, and $\|\ldots\|$ can be defined by letting

$$\|J\| \;=\; \langle\, J,J\,\rangle^{1/2},$$

where $\langle .,.\rangle$ is the inner product. The approximation function $\widehat{J}\,(.,\,r)$ for all possible actions can be seen as an orthogonal projection of J on $\left\{r^{\mathsf{T}}.\,\Phi | r \in \mathbb{R}^{M}\right\}$ with respect to inner product $\langle .,.\rangle$. Therefore, ΠJ is an approximation to J, given the fixed sets of Φ. Van Roy [105] proved (in Theorems 3.9, 3.10, 3.11) that ΠJ is the solution to the least-squares problem

$$r^{*} = \arg\min_{r} \sum_{s\in S} \left[J(s) - \widehat{J}\,(s,r)\right]^{2}. \tag{3.28}$$

Therefore, to update the approximation using TD learning is to perform a gradient decent to update the vector of functional parameters r such that equation (3.28) is solved. r can then be updated after a sequence of observations of size M, where adjustments Δr to r are adopted

$$r \leftarrow r + \sum_{i=1}^{M} \Delta r_{i}. \tag{3.29}$$

One of the main attractions of TD is that learning does not wait until the end of the state sequence to update the approximation, instead, TD represents the complete TD-error δ as a sum of changes in approximation

$$\delta = J - \widehat{J}_{\,t} = \sum_{i=t}^{M} \eta_{i}(\widehat{J}_{\,i+1} - \widehat{J}_{\,i}) \;\text{ where }\; \widehat{J}_{\,M} \stackrel{\text{def}}{=} J. \tag{3.30}$$

Following from equation (3.30), to update r incrementally we expand equation (3.29) as

$$\begin{aligned} r \leftarrow r + \sum_{t=1}^{M} \eta_{t}(J - \widehat{J}_{\,t})\nabla_{r}\,\widehat{J}_{\,t} &= r + \sum_{t=1}^{M} \eta_{t} \sum_{i=t}^{M} (\widehat{J}_{\,i+1} - \widehat{J}_{\,i})\nabla_{r}\,\widehat{J}_{\,t}, \\ &= r + \sum_{i=1}^{M} \eta_{t} \sum_{t=1}^{i} (\widehat{J}_{\,i+1} - \widehat{J}_{\,i})\nabla_{r}\,\widehat{J}_{\,t}, \\ &= r + \sum_{t=1}^{M} \eta_{t}(\widehat{J}_{\,t+1} - \widehat{J}_{\,t}) \sum_{i=1}^{t} \nabla_{r}\,\widehat{J}_{\,i}. \end{aligned} \tag{3.31}$$

Therefore, the incremental adjustment Δr_t to r_t is

$$\Delta r_t = \eta_t(\widehat{J}_{t+1} - \widehat{J}_t) \sum_{i=1}^{t} \nabla_r \widehat{J}_i. \tag{3.32}$$

Δr_t depends on a pair of successive approximations and on the sum of all past values of $\nabla_r \widehat{J}_t$.

3.4.1 TD(λ)

In order to assign greater weight to more recent approximations, TD can be generalised by employing an exponential weighting factor $\lambda \in [0,\ 1]$ in TD as

$$\Delta r_t = \eta_t(\widehat{J}_{t+1} - \widehat{J}_t) \sum_{i=1}^{t} \lambda^{t-i} \nabla_r \widehat{J}_i. \tag{3.33}$$

This generalised representation is referred to as TD(λ). Note that for $\lambda = 1$, TD is equivalent to equation (3.32), referred to as TD(1), where effects on all approximations are taken into account, which is different to when $\lambda = 0$ or TD(0) where Δr is determined only by its effects on the approximation with the most recent observation

$$\Delta r_t = \eta_t(\widehat{J}_{t+1} - \widehat{J}_t)\ \nabla_r \widehat{J}_t. \tag{3.34}$$

In the case of the linear function approximation, TD(λ) for updating of functional parameters can be expressed as

$$r_{t+1} = r_t + \eta_t d_t z_t, \tag{3.35}$$

where d_t is the difference in one-step approximation

$$d_t\ =\ \widehat{J}_{t+1} - \widehat{J}_t, \tag{3.36}$$

and z_t is

$$z_t = \sum_{i=1}^{t} \lambda^{t-i} \phi(s_i). \tag{3.37}$$

Algorithm 7 [15] presents the generalised TD(λ) procedure. The generalised convergence theory of TD(λ) has been presented by Tsitsiklis and Van Roy [15] for discounted ADP with linear function approximation.

Initialisation:

Get parameter vector r_t and learning rate η_t;

Choose λ and set $z_t = \phi(s_t)$.

for *each step t:* **do**

Calculate successive approximations $\widehat{J}(s_t)$ and $\widehat{J}(s_{t+1})$ using

$$\widehat{J}(s,\, r_t) = r_t^{\mathsf{T}}.\, \phi(s),$$

Calculate the difference in approximation

$$d_t = \widehat{J}(s_{t+1}) - \widehat{J}(s_t) = r_t^{\mathsf{T}}.(\phi(s_{t+1}) - \phi(s_t)),$$

Update parameter vector r using

$$r_{t+1} \leftarrow r_t + \eta_t d_t z_t,$$

Update z using

$$z_{t+1} \leftarrow \lambda z_t + \alpha_t \phi(s_{t+1}),$$

$t = t + 1$.

end

Algorithm 7: TD(λ).

Even though TD(λ) is computationally efficient and easy to implement, it does not always converge [15] and has been found to be prone to policy oscillation and overshooting manifested as fluctuating values of r [106, 107], resulting in poor performance. There are also concerns regarding the choice of control parameter η which has to be selected by testing different parameters, and as such this selection is more of an art form.

3.4.2 Least-squares TD learning

To improve the efficiency of TD(λ), particularly with regards to extracting information from experience data, least-squares TD (LSTD) learning was introduced by Bradtke and Barto [108]. After each observation, LSTD solves for parameters r according to the sum of TD errors for all observed data. Recall from equation (3.29) that the functional parameters are the accumulation of all TD updates

$$r \leftarrow r + \sum_{i=0}^{t} \Delta r_i,$$

therefore, the functional parameters r can be found directly from $\sum_{i=0}^{t} \Delta r_i$. Adopting a linear function approximation as in equation (3.10), and let $\mu(r)$ be the sum of all TD updates

$$\mu(r) = \sum_{i=0}^{t} \Delta r_i. \tag{3.38}$$

Furthermore, let $\phi(s) = \phi$ and $g(s_t, u_t, s_{t+1}) = g_t$. Using the δ definition from equation (3.25) and equation (3.10), $\mu(r)$ can be rearranged as

$$\begin{aligned} \mu(r) = \sum_{i=0}^{t} \phi_i \delta_i &= \sum_{i=0}^{t} \phi_i (g_i + \alpha r^{\mathsf{T}} . \phi_{i+1} - r^{\mathsf{T}} . \phi_i), \\ &= \sum_{i=0}^{t} (\phi_i g_i \; - \; \phi_i (\phi_i - \alpha_{i+1} \phi_{i+1})^{\mathsf{T}} . r), \\ &= \sum_{i=0}^{t} \phi_i g_i - \sum_{i=0}^{t} \phi_i (\phi_i - \alpha_{i+1} \phi_{i+1})^{\mathsf{T}} . r. \end{aligned} \tag{3.39}$$

It can be seen that equation (3.39) has the form

$$\mu(r) = b_t - \mathbf{A}_t r,$$

and given that LSTD chooses r such that the sum of TD updates is zero, we solve for the new parameters using

$$r_{t+1} = \mathbf{A}_t^{-1} b_t. \tag{3.40}$$

Initialisation:

Initialise r_0 arbitrarily;

Choose initial state s_0 and set $\mathbf{A}_0 = 0$, $b_0 = 0$.

for *each step t:* **do**

Receive one-step cost g_t and next state s_{t+1},

Update b

$$b_t \leftarrow b_{t-1} + \phi(s_t) g_t,$$

Calculate difference in ϕ

$$d_t = \phi(s_t) - \alpha_{t+1} \phi(s_{t+1})$$

Update $\mathbf{A}$

$$\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \phi(s_t) d_t^{\mathsf{T}},$$

Calculate new parameters r

$$r_{t+1} = \mathbf{A}_t^{-1} b_t,$$

$t = t + 1$.

end

Algorithm 8: Least-squares TD learning.

To implement LSTD for real-time control, we incorporate each observed cost and state transition into matrix $\mathbf{A}$ and vector b, then solve for a new parameter vector r. It should be noted that once $\mathbf{A}$ and b are updated, the experience can be overlooked without losing information.

By incorporating all temporal updates, LSTD fully exploits the observed data from the start of the optimisation process.

In general, LSTD uses past information more fully than does TD(λ), but due to the requirement of inverting a matrix in equation (3.40), it is more demanding computationally. LSTD has no requirement for control parameters η, thus reducing the chances of poor performance arising from an unfortunate choice of control parameters. Algorithm 8 [108] presents a generalised LSTD learning framework.

Similar to TD(λ), LSTD can be adopted to assign greater weight to more recent approximations using an exponential weighting factor λ, which is referred to as LSTD(λ) [109].

3.5 Newton-based TD learning

Recall from Section 3.4 that TD learning is concerned with minimising the temporal-difference error δ

$$\delta_t = g(s_t, u_t, s_{t+1}) + \alpha_t \widehat{J}\,(s_{t+1}, r_t) - \widehat{J}\,(s_t, r_t),$$

by iteratively solving the least-squares problem

$$r^* = \arg\min_r \sum_{s \in S} \left[J(s) - \widehat{J}\,(s, r)\right]^2.$$

A different method for finding adjustments Δr to r, compared to the previous sections, is to use the Newton's method to find the root of δ in the variable r, *i.e.* $\delta(r) = 0$.

Using Newton's method, given some point x_k, we may estimate the root of function $f(x)$,

by constructing the tangent to the curve of $f(x)$ at x_k and noting where that linear function is zero. Clearly for Newton's method to be defined we need $f(x)$ to be differentiable, otherwise the tangent may not exist. For a single variable function, starting from an initial point x_0 we can construct the linear approximation of $f(x)$ in the neighbourhood of

$$x_0 : f(x_0 + h) \approx f(x_0) + f'(x_0)h,$$

and solve the arising linear equation

$$f(x_0) + f'(x_0)h = 0.$$

Thus, we arrive at recurrent method

$$x_{k+1} = x_k - f'(x_k)^{-1} f(x_k). \tag{3.41}$$

To extend the Newton's method to n variables, the iteration for $\min_x f(x)$ becomes

$$x_{k+1} = x_k - \mathbf{H}(x_k)^{-1} f'(x_k), \tag{3.42}$$

where $\mathbf{H}$ is the Hessian matrix of $f(x)$.

Now, as with the standard TD literature by adopting the quadratic error measure

$$e_t(r) = \delta_t(r)^2, \tag{3.43}$$

a Newton-based TD learning calculates the M parameter values that would minimise equation (3.43) when aggregated over past steps by solving after each observation

$$\min_r E_t(r) = \sum_{i=0}^{t} \beta_i \, e_{t-i}(r), \tag{3.44}$$

for some positive discounting factors $\beta_i (0 \leq i \leq t)$ that are non-increasing in i . By decreasing rapidly with i, these discounting factors can be used to weight recent discrepancies more heavily with the effect that the objective function will emphasise the corresponding operating conditions. Since the minimum of equation (3.44) is the global minimum, we can look for the the points where $\nabla E(r) = 0$, and therefore, the Newton-based solution to equation (3.44) can be used to calculate adjustments Δr to r that would minimise the current aggregated discrepancy as:

$$\Delta r_t = \arg\min_{\Delta r} E(r_t + \Delta r). \tag{3.45}$$

For the solution to equation (3.45) to be well-defined, $\beta_{M-1} > 0$ so that there are at least as many contributions $e_i(r)$ to the minimand as M , the number of parameters in r .

In the case that additionally $\beta_M = 0$ and the stationarity conditions for minimality of the solution are mutually independent, there will be a unique solution of equation (3.45) to equation (3.44).

In the case $\beta_{M-1} = 0$, there will be a continuum of solutions. In such cases, adjusting to $r_{t+1} = r_t + \Delta r_t$ has been found to be prone to policy oscillation and overshooting manifested as fluctuating values of r, and consequently, when using only current observations (*i.e.* $\beta_1 = 0$), adjustments can be moderated similar to TD(λ) using equation (3.14)

$$r_{t+1} = r_t + \eta_t \Delta r_t,$$

where η_t satisfies the conditions for convergence of equation (3.15)

$$\sum_{t=0}^{\infty} \eta_t = \infty, \sum_{t=0}^{\infty} {\eta_t}^2 < \infty.$$

In this approach, use of rapidly decreasing discounting factors $\beta_i (0 \leq i \leq t)$ that emphasise recent discrepancies will lead to greater adaptiveness of the ADP optimisation of equa-

tion(3.7)

$$\widetilde{J}\,(s_t, r_t) = \min_{u_t \in U} E\left\{ g(s_t, u_t, s_{t+1}) + \alpha_{t+1} \widehat{J}\,(s_{t+1}(u_t), r_t) \right\},$$

to current operating conditions. Similarly, greater values of η_t allow for greater adjustment to the parameters r so can lead to more rapid adaptation of the system but expose it to stochastic fluctuations, whilst lesser values of η_t confer greater stability on the values of parameters r .

From equation (3.42), for the cases where $\beta_{M-1} > 0$, the following Newton-based adjustment can be applied once after each optimisation:

$$\Delta r_t = -\left(\sum_{j=0}^{t} \beta_j \, \nabla \delta_{t-j}(r_t) \, \nabla \delta_{t-j}(r_t)^{\mathsf{T}} \right)^{-1} \sum_{i=0}^{t} \beta_i \, \delta_{t-i}(r_t) \nabla \delta_{t-i}(r_t) \, , \tag{3.46}$$

using the δ definition from equation (3.25) and equation (3.10)

$$\delta_t = g(s_t, u_t, s_{t+1}) + \alpha_{t+1} r_t^{\mathsf{T}} . \phi(s_{t+1}) - r_t^{\mathsf{T}} . \phi(s_t). \tag{3.47}$$

The optimisation from equation (3.44) is then quadratic in r and is solved exactly, where it can be calculated, by equation (3.46) which becomes

$$\Delta r_t = -\mathbf{H}_t^{-1} \sum_{i=0}^{t} \beta_i \, \delta_{t-i}(r_t) \varphi_{t-i} \, , \tag{3.48}$$

where

$$\mathbf{H}_t = \sum_{j=0}^{t} \beta_j \, \varphi_{t-j} \varphi_{t-j}^{\mathsf{T}}, \text{ and } \varphi_k = \alpha_{t+1} \phi(s_{t+1}) - \phi(s_t).$$

The initial value r can be an arbitrary vector, or calculated according to a period of training using historical data.

Certain special cases of TD learning arise according to the specification of β_i that are worthy of note. We consider these in turn.

Initialisation:

Initialise r_0 arbitrarily and η_0;

Choose initial state s_0 and set the Hessian matrix $\mathbf{H}_0 = 0$.

for *each step t:* **do**

Receive one-step cost g_t and next state s_{t+1},

$\varphi_t = \alpha_{t+1}\phi(s_{t+1}) - \phi(s_t)$,

if $t < M-1$ *:* **then**

Update $\mathbf{H}$ using

$$\mathbf{H}_t = \mathbf{H}_{t-1} + \varphi_t \varphi_t^{\mathsf{T}} ,$$

else

Update $\mathbf{H}$,

Calculate the temporal-difference δ_t

$$\delta_t = g(s_t, u_t, s_{t+1}) + \alpha_{t+1}\widehat{J}(s_{t+1}, r_t) - \widehat{J}(s_t, r_t),$$

Update r

$$r_{t+1} = r_t - \eta_t \left(\mathbf{H}_t^{-1} \delta_t(r_t)\varphi_t \right) ,$$

end

$t = t+1$.

end

Algorithm 9: One-step Newton-based TD learning.

3.5.1 One-step Newton TD learning

For one-step Newton TD learning, $\beta_1 = 0$ in equation (3.48). In this case, only the current discrepancy δ_t is considered in calculating Δr_t. This will lead to degenerate solutions whenever $M > 1$ because the matrix $\nabla_r^2 \delta_t$ of second derivatives will be singular and hence

non-invertible: there will be a continuum of solutions r to $\delta_t(r) = 0$ that will minimise δ_t^2 to 0. In such cases, a suitable direction Δr_t can be furnished by the gradient as

$$-\nabla_r \delta_t = -(\alpha_{t+1}\phi_{t+1} - \phi_t),$$

with modification factor η_t calculated either to achieve $\delta_t(r_t - \eta_t \nabla_r \delta_t) = 0$ or some proportion of this. This can be solved using the accumulated estimate $\mathbf{H}_t$ of the Hessian matrix $\mathbf{H}$ in a single Newton step:

$$\Delta r_t = -\mathbf{H}_t^{-1}\ \delta_t(r_t)\varphi_t \text{ where } \mathbf{H}_t = \frac{1}{t}\sum_{j=0}^{t}\ \varphi_{t-j}\varphi_{t-j}^{\mathsf{T}}\ . \tag{3.49}$$

3.5.2 Newton LSTD learning

Recall from Section 3.4.2, after each observation, LSTD solves for parameters r by minimising the sum of squares of all temporal differences $\delta_i (0 \leq i \leq t)$ since the start of operation. In this case, where $\beta_i = \frac{1}{t}$, the objective

$$E_t(r) = \frac{1}{t}\ \sum_{i=0}^{t}\ \delta_{t-i}^2(r)\ , \tag{3.50}$$

of equation (3.44) is quadratic in r and when $t \geq M-1$ can be solved in a single Newton step. At this solution,

$$\nabla_r E_t(r_t) = 0\ (t \geq M-1)\ . \tag{3.51}$$

Thus for $t \geq M$ the solution to the minimisation of equation (3.44) is achieved by the single Newton step specified by equation (3.48). But from equation (3.50)

$$E_t(r) = \Big(\frac{t-1}{t}\Big)E_{t-1}(r)\ +\ \frac{1}{t}\delta_t^2(r)\ (t \geq M). \tag{3.52}$$

Therefore Δr_t is

$$\Delta r_t = \ \frac{-1}{t}\mathbf{H}_t^{-1}\delta_t(r_t)\varphi_t\ (t \geq M), \tag{3.53}$$

where

$$\mathbf{H}_t = \sum_{j=0}^{t} \frac{1}{t} \varphi_{t-j} \varphi_{t-j}^{\mathsf{T}} .$$

Initialisation:

Initialise r_0 arbitrarily;

Choose initial state s_0 and set the Hessian matrix $\mathbf{H}_0 = 0$.

for *each step t:* **do**

Receive one-step cost g_t and next state s_{t+1},

$\varphi_t = \alpha_{t+1}\phi(s_{t+1}) - \phi(s_t)$,

if $t < M-1$ *:* **then**

Update $\mathbf{H}$ using

$$\mathbf{H}_t = \mathbf{H}_{t-1} + \frac{1}{t} \varphi_t \varphi_t^{\mathsf{T}} ,$$

else

Update $\mathbf{H}$,

Calculate the temporal-difference δ_t

$$\delta_t = g(s_t, u_t, s_{t+1}) + \alpha_{t+1} \widehat{J}\,(s_{t+1}, r_t) - \widehat{J}\,(s_t, r_t),$$

Update r

$$r_{t+1} = r_t - \frac{1}{t}\left(\mathbf{H}_t^{-1}\, \delta_t(r_t)\varphi_t\right) ,$$

end

$t = t+1$.

end

Algorithm 10: Newton-based LSTD learning.

.

This is a proportion $(1/t)$ of the one-step TD change given in equation (3.49), so with diminishing influence of subsequent discrepancies δ_t as t increases.

3.5.3 Exponentially weighted Newton TD learning

In the case where $\beta_{i+1} = \gamma\beta_i$ $(0 \leq i \leq t)(0 \leq \gamma \leq a)$, γ can be used to attach reduced weight to discrepancies in the past, which were assessed using earlier estimates of the parameters r and so will be less reliable. The case of one-step TD learning can be represented by $\gamma = 0$ and that of LSTD by $\gamma = 1$.

When $\gamma \in (0,1)$, the objective of the ADP optimisation of equation (3.7)

$$\widetilde{J}(s_t, r_t) = \min_{u_t \in U} E\left\{ g(s_t, u_t, s_{t+1}) + \alpha_{t+1}\widehat{J}(s_{t+1}(u_t), r_t) \right\},$$

is equivalent to the exponentially weighted moving average $E_t(r)$ of the squared discrepancies:

$$E_t(r) = (1-\gamma)\sum_{j=0}^{t} \gamma\delta_{t-j}^2(r) \,. \tag{3.54}$$

The resulting optimisation of equation (3.44) is again quadratic in r and when $t \geq M-1$ can be solved in a single Newton step.

An analysis corresponding to that for LSTD can be applied here because

$$E_t(r) = \gamma E_{t-1}(r) \;+\; (1-\gamma)\delta_t^2(r) \;(t \geq M). \tag{3.55}$$

Thus, the update to the parameters r at step t is the constant proportion $(1-\gamma)$ of the one-step TD change of equation (3.49)

$$\Delta r_t = -(1-\gamma)\mathbf{H}_t^{-1}\delta_t(r_t)\varphi_t \text{ where } \mathbf{H}_t = \sum_{j=0}^{t} \beta_j \varphi_{t-j}\varphi_{t-j}^{\mathsf{T}} \,. \tag{3.56}$$

3.6 Summary and discussion

Dynamic programming (DP) provides an elegant way of expressing sequential decision making problems, by breaking them down into smaller sub-problems that can be solved more effectively. Even though DP is a powerful method, it may become computationally intractable because it requires looping over the entire space, decision and information spaces to calculate the value function which is the long term cost resulting from making a decision; this is referred to as DP's *curses of dimensionality*, and renders DP undesirable for real-time (or low-latency) applications. Beyond this, the requisite information in th future will not wholly be available to inform decisions.

To address these difficulties with DP, the *approximate dynamic programme* (ADP) approach to decision-making has been developed. This approach works forward in time and employs a greedy approach to make decisions using available information in an exhaustive search with explicit evaluation in the short-term together with an approximation to the value function after that. This means that the ADP does not compute the value function explicitly, and so avoids the need to solve the *Bellman's* equation for each state at each future step. Therefore, we reduce the future extent of the state space considered by replacing the true value function with an approximation, normally constructed using a parameterised function approximator, and do not represent future decisions explicitly. This has the effect to make the computational requirement polynomial to the number of state variables, rather than being exponential to the size of state space.

To improve approximations, *reinforcement learning* (RL) methods can be utilised to learn from experience, similar to the process of learning in intelligent beings. Learning from interaction is the foundation of RL, where an *agent* interacts with the environment on each step, using an available decision and then observes the cost incurred from that decision. The idea is to discover which decisions yield the least cost (or most reward) by trying many decisions randomly and observing their outcomes.

RL can also be used to tackle problems where a model of the environment may not be captured analytically due to the complexity of the system. In this chapter, popular RL methods,

Q-learning, *SARSA* and the *actor-critic* algorithms were presented which can be used to solve problems without a model of the environment. In the present case of railway traffic management, given that a model of the environment is available, the focus is on RL as a learning method for ADP.

A widely used RL method is *temporal-difference* (TD) learning and it is central to many RL and ADP implementations. TD is a powerful learning method which updates the value function at each step, by comparing the estimation of the long term cost and the actual observed value of the long term cost, to construct a *TD-error*. The idea is to minimise this error iteratively as more experience is accumulated over time.

Specific TD learning methods, TD(λ) and LSTD, perform a gradient decent on the TD-error with respect to the approximation parameters to minimise the error in approximation. Another method is to use the Newton's method to find the root of the TD-error with respect to the approximation parameters, which we refer to as Newton-based TD learning. Out of these, TD(λ) is the most computationally efficient, however, it is found in some cases to be prone to overshooting and policy oscillation which result in fluctuations in approximation and therefore poor performance. Therefore, TD(λ) requires assignment of control parameters which require prior tuning for implementation.

Newton-based TD learning and LSTD learning are computationally more demanding, except in cases were low dimension approximation functions are used. However, they do not require control parameters, reducing the chances of poor performance arising from a poor choice of values. They also use past information more fully than does TD(λ). It is easier to control adjustments to the approximation function in Newton-based TD compared to LSTD, therefore, Newton-based TD may provide more control on the adaptivity of ADP.

There are many learning methods that are not presented in this chapter but can be adopted and utilised in ADP. The choice of learning is always domain-specific. In some cases the computational time is of paramount importance, in others more accurate approximations are needed with less regard to the computational time. In this thesis, the primary objective is to control railway traffic at isolated critical locations (*e.g.* at junction) in a timely manner;

even though DP is not practical to solve this problem in real-time, pure RL methods such as *SARSA*, though capable of tackling complex problems, may also not be as appropriate as ADP because the model of the system can be used to better inform decision making.

In the next chapter, we describe the dynamics of railway traffic management, and develop the present system based ADP and discuss practical requirements for real-time operations.

Chapter 4

Adaptive Railway Traffic Control

"A small body of determined spirits fired by an unquenchable faith in their mission can alter the course of history." - Mahatma Gandhi

Current and planned railway control systems and railway traffic management (RTM) procedures were reviewed in Chapter 2, and Chapter 3 presented the fundamentals of approximate dynamic programming (ADP) and reinforcement learning (RL). In this Chapter, the application of ADP for railway traffic control is presented, therefore bringing together Chapters 2 and 3. At first, the state-space of the RTM formulation as used for optimisation in this thesis is discussed, then the procedure that solves our RTM problem using ADP by employing the techniques derived in Chapter 3 is set out. Thus, the control method proposed here can adapt according to prevailing traffic conditions on the railway network and calculate decisions accordingly. Moreover, system architecture of the adaptive railway traffic controller (ARTC) is explained by presenting the data flow in the system and the requirements associated with them. Finally, the realisation of ARTC in computer simulations is discussed towards the end of this Chapter. All systems described here are implemented in Python programming language.

4.1 Control framework

The primary objective of this thesis is to develop a method to control railway traffic in real-time and optimise capacity utilisation in events of deviation of operation from the timetable. To this end, re-ordering or re-timing of trains at critical points such as junctions, crossings and merges present opportunities in terms of recovering deviations from the timetable or avoiding further deviations. Other possible actions to optimise capacity include re-routing of trains, and platforming at stations.

In this Section, we focus first on representing the re-scheduling problem of re-ordering (or sequencing) at critical points using an event-based traffic model, similar to the one used in the work of Ho et al. [5], and formulate this as a dynamic programme; then we adopt ADP with the Newton-based TD learning methods of the previous Chapter to build an adaptive framework for railway traffic control, and discuss how this framework can be readily extended to include actions such as re-routing and platforming.

4.1.1 Dynamic programming formulae

As was described in Chapter 2, deviations from the timetable may result in further perturbations on the rail network as delays propagate to other trains, mostly due to signalling constraints put in place to ensure safe operations. Therefore, minor deviations magnify and result in services not arriving at critical locations, where the right of way is not exclusive to one direction of traffic, at the scheduled time.

One significant critical point on railway networks are junctions where traffic from more than one direction merges. The sequencing of trains at a railway junction is therefore, a schedule that includes a sequence of right-of-way assignments which optimises the junction capacity according to a specific objective. Thus, traffic control at a railway junction can be regarded as a multistage decision-making process.

In such a process, the state-space of the problem can be represented as stages, characterised by the number of trains remaining in the system awaiting sequence assignment, and passage

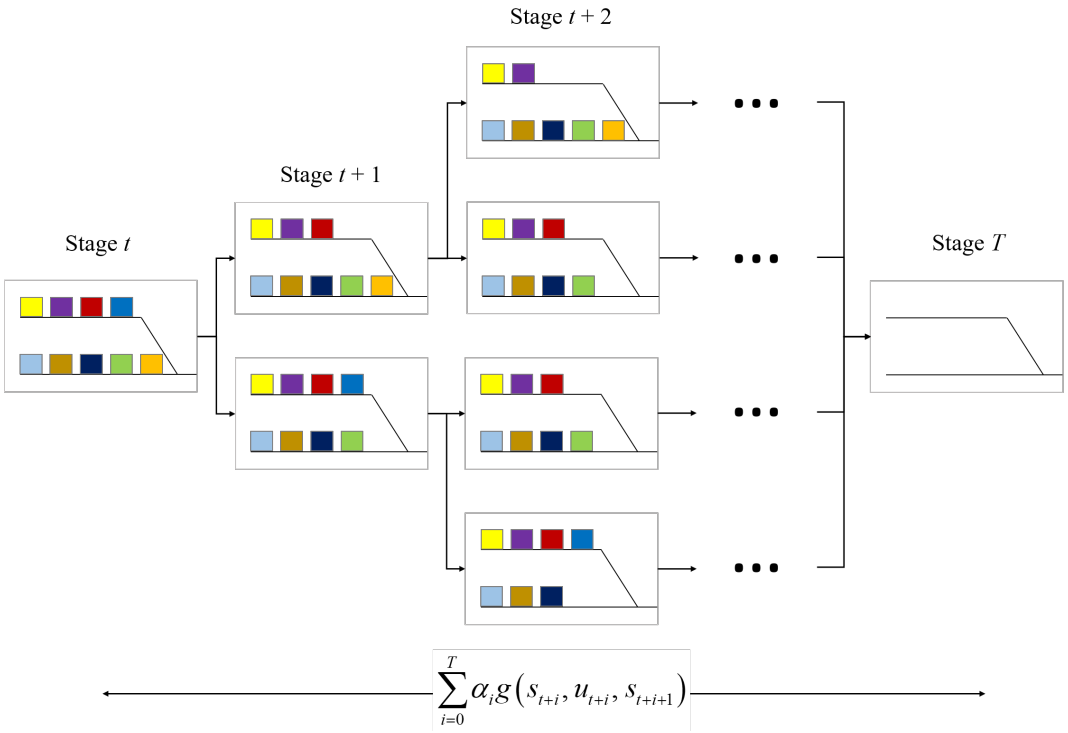


Figure 4.1: DP representation of the state-space for nine trains.

of one train through the junction is considered as transition from one stage to the next. Figure 4.1 illustrates the state-space for nine trains for a sequencing problem at a single railway junction where stages depend on the number of trains present in the system.

According to Figure 4.1, control actions are assignment of right-of-way to one of the converging routes, and different control actions result in different speeds and positions of the remaining trains. It follows that the states are defined not only by the number of trains, but also by their associated speeds and locations, so that presence of the same combination of trains in a stage of Figure 4.1, does not necessarily represent the same state.

Excluding the initial stage, the system should undergo as many stages in the future as the number of trains currently in the control area. The number of possible states in each stage expand rapidly until an intermediate stage is reached, after which the number of possible states decrease. Dynamic programming can be used to solve exactly for the optimal decision for each stage.

Therefore, the optimal solution to Figure 4.1 is a sequence of rights-of-ways for each stage

that minimises a desired objective function from stage t to stage T. Recalling definitions from Chapter 3, the dynamic programme is then to solve

$$\min_{\pi \in U} E\left\{ \sum_{i=0}^{T} \alpha_i \, g(s_{t+i}, u_{t+i}) \mid s_0 \right\}, \tag{4.1}$$

and the optimal performance J starting from state s_t at step t can be calculated by solving Bellman's recursive equation

$$J(s_t) = \min_{u \in U} E\left\{ g(s_t, u) + \alpha_{t+1} J(s_{t+1}(u)) \right\}. \tag{4.2}$$

Here, the one-step cost g represents the cost of transitioning from state s_t to s_{t+1} by choosing action u_t. The exact form of 'cost' can be selected according to requirements of practitioners and their definitions of delays. For instance the objective can be to minimise the *schedule 8* cost incurred by Network Rail, the infrastructure manager in Great Britain, which compensates train operators for unplanned service disruption caused by infrastructure asset failures. The choice of objective function is discussed in more detail in Chapter 5, where variants of delay are minimised.

Figure 4.1 illustrates the extent of stages that g is required to be calculated over, and it is evident that as the number of trains grows, the computational effort needed to calculate $J(s_t)$ for all $s \in S$ renders the problem progressively larger. As RTM is a real-time system, even consideration of few trains could be impractically time consuming.

4.1.2 Adaptive control with TD learning

The essence of ADP is to focus computational effort on the short term where the information is more reliable and decisions imminent by replacing the long term performance with an approximation.

To adopt the state-space representation of Figure 4.1 for ADP, we follow a short planning horizon T which is evaluated explicitly, and follow an approximation for evaluation of per-

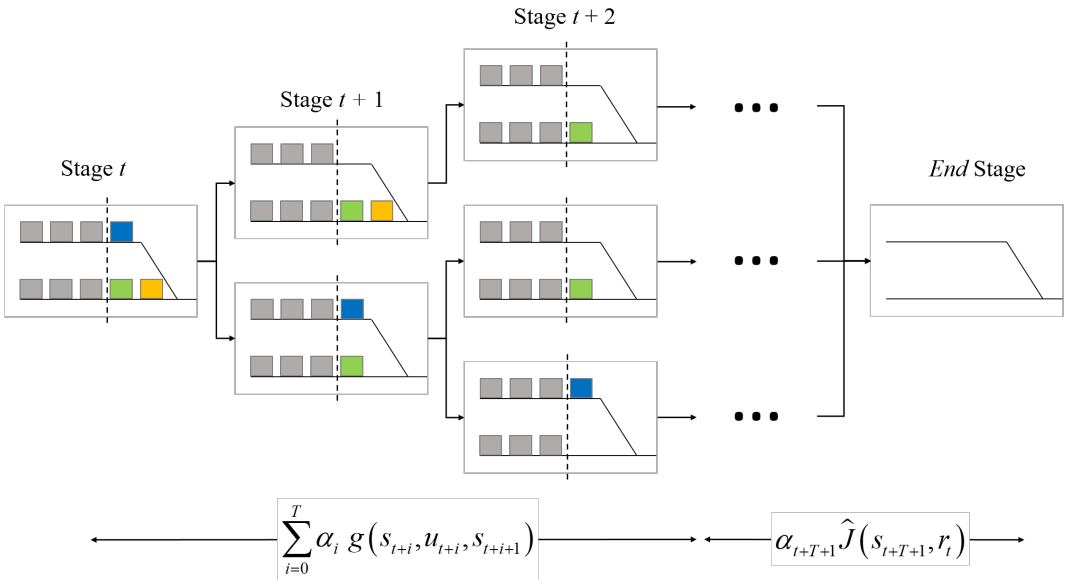


Figure 4.2: Representation of the state-space in ADP with a horizon of three trains *i.e.* $T = 3$.

formance after the planning horizon T. Figure 4.2 illustrates what this means in terms of decision making for a horizon of three trains.

Comparison of Figures 4.1 and 4.2 reveal the true extent of computational efficiency of ADP for RTM. Whereas, in Figures 4.1, most stages up to the final one need to be evaluated explicitly to compare candidate decisions, in Figure 4.2 only the first few stages are evaluated explicitly (regulated by the choice of T) and performance by the later stages is approximated, resulting in savings in computational effort. Figure 4.2 also illustrates the importance of good approximations, as ARTC relies on them to calculate decisions that are appropriate for long-term performance.

The availability of real-time data and the stochastic nature of railway operations are a few potential barriers to accurate anticipation of future traffic which in turn would effect the quality of decisions produced by the RTM system. In this study it is assumed that sufficient real-time data are available in railway systems for the near future to facilitate effective anticipation of future traffic, and also other data are available that describe the probability of events occurring in operations, *e.g.* distributions for dwell times at stations.

As new data emerge in real-time, the quality of traffic anticipation will improve, and according to TD learning, as more traffic outcomes of various decisions (or state-action pairs)

are observed the approximations are expected to improve as well. Therefore, a rolling horizon approach is adopted in this framework, where only the first part of decisions (*i.e.* the decision for the next stage) is implemented and ARTC re-calculates a new sequence of decisions for the next time horizon T. Figure 4.3 presents the implementation of the rolling horizon approach with a horizon of T stages within the present ADP framework.

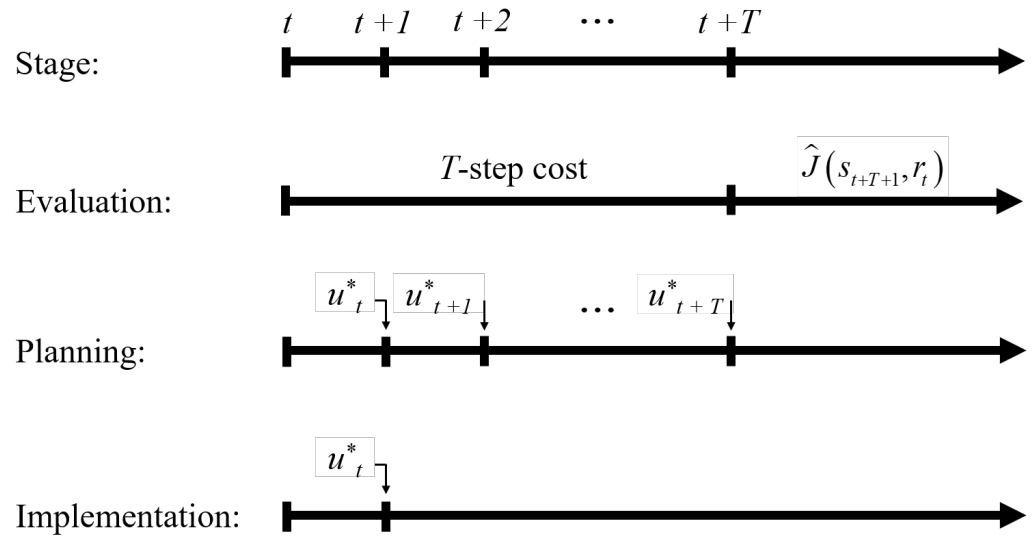


Figure 4.3: Control policy for adaptive railway traffic control with rolling horizon implementation.

To achieve real-time sequence control of trains at junctions, the general representation of ADP presented in Chapter 3 is therefore applied to control railway traffic, by considering decisions u_t at stage t over horizon of T stages to be considered explicitly. The resulting interpretation of Bellman's equation is then

$$\widetilde{J}(s_t) = \min_{\pi \in U} E\left\{\left[\sum_{i=0}^{T} \alpha_i g(s_{t+i}, u_{t+i})\right] + \alpha_{T+1}\widehat{J}(s_{t+T+1}, r_t)\right\}. \tag{4.3}$$

A sequence of decisions π_t^*, consisting of a sequence of decisions *i.e.* $(u_t, u_{t+1}, ..., u_T)$, is therefore calculated by solving

$$\pi_t^* = \arg\min_{\pi \in U} E\left\{\left[\sum_{i=0}^{T} \alpha_i g(s_{t+i}, u_{t+i})\right] + \alpha_{T+1}\widehat{J}(s_{t+T+1}, r_t)\right\}, \tag{4.4}$$

where each stage cost $g(s_i, u_i, s_{i+1})$ represents a form of 'cost' measurement incurred by all trains under consideration within the control area.

In the present case of railway traffic control, the state s_t is a combination of traffic state and control state at stage t, and the controls π specify the sequence of mutually incompatible trains at relevant places in the control area. As was discussed, only the first of the calculated decisions (*i.e.* $u_t^* \in \pi_t^*$) is implemented before incrementing t and recalculating equation (4.4) at the next stage.

In this framework, a linear form for the approximation function $\widehat{J}(s,r)$ is employed, and a temporal difference δ_t at stage t is defined by

$$\delta_t = \left[\sum_{i=0}^{T} \alpha_i g(s_{t+i}, u_{t+i})\right] + \alpha_{T+1}\widehat{J}(s_{t+T+1}, r_t) - \widehat{J}(s_t, r_t). \tag{4.5}$$

The procedures for Newton-based TD learning to calculate adjustments Δr_t to r_t, are then as described in Algorithms 9 and 10 of Chapter 3.

As was discussed in Section 4.1.1, system states are defined not only by the number of trains, but also by their associated speeds and locations. Therefore, each decision in Figure 4.2 may result in a different set of extracted state features ϕ and therefore a different structure for the value function approximation $\widehat{J}$. This means a different $\widehat{J}$ when different trains remain in the control area after implementation of π_t^*, but also, in states after the blue train and the green train leave the control area in Figure 4.2. Furthermore, different decisions u could have a different effect on speed and position of the same train and for instance the two states after the green train leaves the control area in Figure 4.2 may also result in different $\widehat{J}$ as different states can be expected to lead to different $\widehat{J}$ values. According to this, the extracted features ϕ play a central part in the approximation function and the learning framework is sensitive to the choice of features, as are all reinforcement learning methods.

4.2 System architecture

Up to now, this thesis has focused primarily on developing a methodology based on ADP and RL to address RTM. This methodology is envisaged for real-time operations and would require low-latency communication with other railway systems to produce revised schedule for trains. Therefore, the adaptive framework described in this Chapter is a system that requires integration with other systems to realise its purpose.

This Section describes the system architecture of the adaptive railway traffic controller formulated in the previous Section. Different components developed to realise the controller are described in this Section as well as requirements to allow the system to function as intended.

The aim of a traffic management system (TMS) is to anticipate conflicts resulting from deviations from the timetable and alleviate delays by managing arrival/departure times at conflict points or through re-routing of trains, if such action is possible and permitted. These adjustments can then be communicated to trains via the line-side colour light signalling system or via on-board Driver Machine Interface (DMI) in the form of an updated speed profile or change in the Limit of Movement Authority. In the re-scheduling literature, these functions form different parts of the RTM system and are referred to as conflict anticipation, conflict resolution and plan coordination respectively.

The real-time RTM system will ideally solve for a conflict free schedule for all trains in a designated control area, that maximises the capacity utilisation of the network under consideration. The plans need to be compatible with operational requirements of the railway network and must respect railway safety constraints. This should be done using real-time data from all trains and with consideration of knock-on effect of the schedules on other trains and traffic that will be affected downstream.

In the present adaptive framework, as is required for calculation of the one-step cost g in equation (4.4), there is need for anticipation of the near-future. This overlaps with the conflict anticipation function of the RTM system, the difference being more frequent calling of this function in ARTC. Furthermore, plan coordination requires feasibility check of the new

schedules to inform trains of their planned arrival and departure times at signals, stations and the conflict points. To perform traffic anticipation and plan coordination, a high fidelity microscopic railway simulator is developed for this research, which is described in Section 4.2.2.

Given the definitions of the adaptive procedure set out in Section 4.1.2, the framework is designed as an event-based system where entry of a train into a designated area and passage of a train through the conflict point would trigger optimisation of further trains sequences at the conflict point. The planned sequence is then coordinated using the traffic simulator to derive detailed timings on the entry of every block section in the control area. This detailed plan may also be beneficial for other envisaged optimisation systems such as trajectory optimisation for efficient energy usage.

The resulting ARTC could be implemented either as a decision support tool for signallers or as a fully automated system in charge of making scheduling decisions within a railway network. In either of these implementation scenarios, the fundamental requirements and

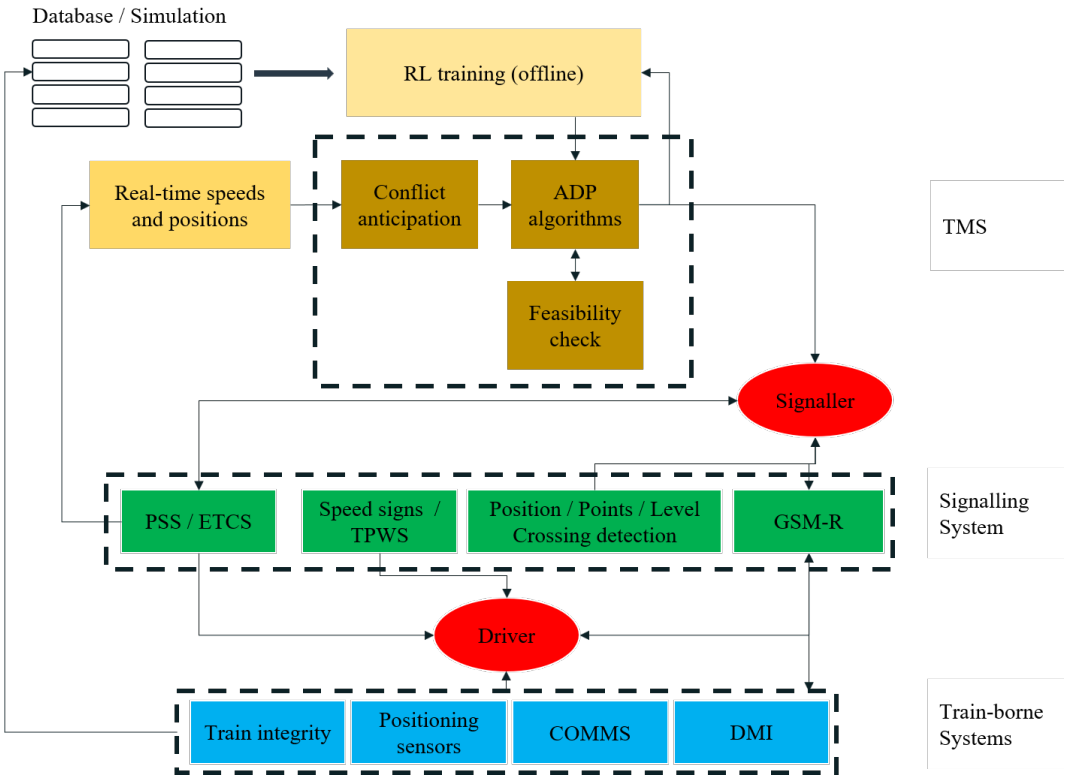


Figure 4.4: RTM with ADP system architecture for re-scheduling support.

dependencies of ARTC to operate remain the same, be it with different safety integrity level specifications. In both options, ARTC requires good quality real-time dynamic data (data that change over time) for all trains as well as other static data (data that do not change over time).

Figure 4.4 presents ARTC system architecture as a decision support system and Figure 4.5 presents ARTC system architecture for automated traffic control. The main difference between these implementation scenarios is human intervention in carrying out suggested optimised plans. One area where human intervention needs to be considered, as opposed to ARTC, is the signaller workload. If the traffic management system (TMS) is implemented as a decision support tool, then there are limitations to how many changes a signaller can make to signalling plans without being overworked, which could increase the propensity to mistakes and potential benefits not realised. Driver behaviour may also play a part in realising benefits from ARTC, as plans may need to be altered if a train is not travelling as was anticipated.

From a system point of view, there are no additional requirements to those of conventional

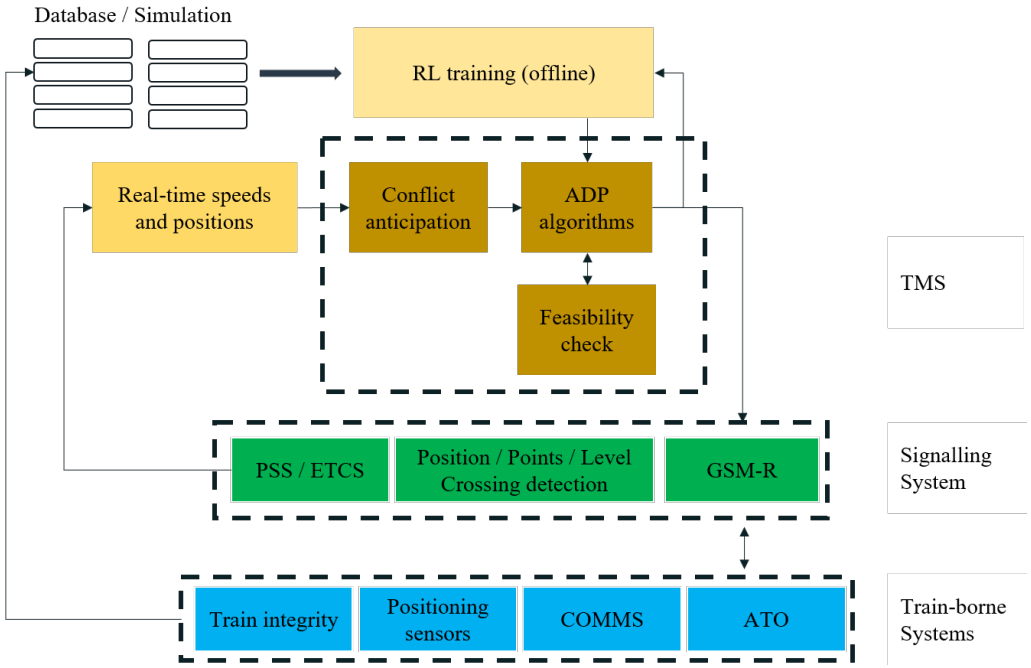


Figure 4.5: RTM with ADP system architecture for automated traffic control.

signalling system, though sensors are required on-board trains to report accurate speed and position information; this functionality is already met by planned ETCS levels.

Real-time traffic optimisation will only be possible with the help of emerging communication and speed/positioning systems. However, there are issues associated with the use of such systems. All sensors, including on-board position sensors, are prone to errors that affect the confidence in their measurements, and even failure. The aim of locating an object, in any application, is to know its position accurately. When speed and location sensors are used in the railway sector, the accuracy and the degree of confidence of the provided position are the main concerns. Also, there are communication delays associated with the use of railway communication systems. For instance, the time taken for data to be collected by a train and sent to a railway control centre can take up between 2 to 7 seconds alone; and the communication between a control centre and the train can take up to 8-9 seconds [110]. This means by the time data is collected by a train, sent to the control centre, and new train control decisions are sent back to trains, the trains would have travelled forward by some distance: at speeds of 225 km/h, this could be as much as 1 km.

In this thesis it is assumed that sufficient and good quality real-time data are available by railway systems for the near future as to facilitate effective anticipation of future traffic, *e.g.* good quality arrival times into the control area, and other data are available that describe the probability of events occurring in operations, *e.g.* distributions for dwell times at stations. Furthermore, it is assumed that no latency is experienced by the system to receive data, calculate plans and deliver optimised sequences to the signaling system, *i.e.* no communication and computation delays. The effect of these idealised assumptions will be to provide a bound on the limits of performance that could be achieved by systems of this kind.

4.2.1 Real-time control procedure

As was noted briefly in the previous section, ARTC requires static and dynamic data, for the **control area** only, to perform traffic optimisation. Static or off-line data are those that will not vary at any time during the planned train operations and include:

- Infrastructure data: the infrastructure consists of a set of available block sections with associated signals. For each block section the occupancy status, length, gradient, speed limitations, directions and maximum speeds are required. Furthermore, locations of junctions, stations and level crossings need to be specified.

- Train characteristics: the specific technical characteristics of the rolling stock (power, train length and mass) are required.

- Signalling system: signalling system type (*e.g.* conventional multiple-aspect or ETCS), signalling aspects, route release and switching times are included in this data.

- Timetable: the timetable contains a list of arrival and departure times (time windows of minimum/maximum arrival/departure times) for a set of relevant points in the network, including all the station platform tracks visited by each train, and a list of routes for each service. Other station related data are included here, *e.g.* distributions for dwell times at stations.

The acceleration and braking rates are to be calculated on the basis of traction force and scheduled maximum speeds. It is assumed that the driver behaviour is known in advance.

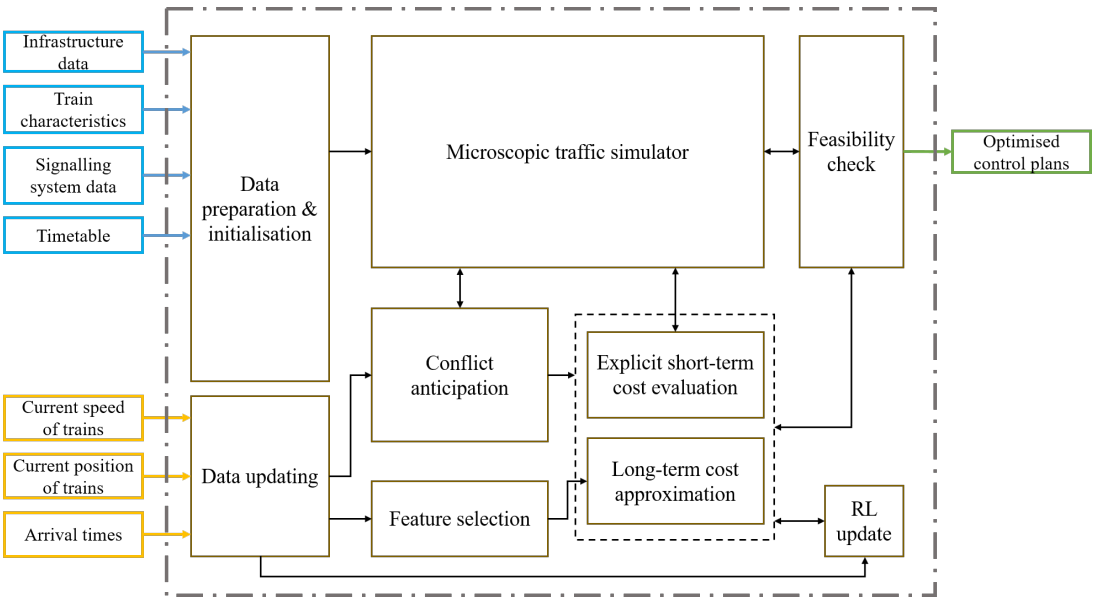


Figure 4.6: ARTC real-time control procedure.

Furthermore, the weather condition, the train load (number of passengers) and weight are assumed to be known in advance.

Dynamic data are data that change in the course of train operations and need to be updated and transmitted to the optimiser in real-time or dynamically. They include:

- Current speed of each train within the control area.
- Current position of each train within the control area.
- Arrival times of each train at key points within the control area: entry into the control area, at conflict points, at stations, and at exits from the control area.

In real-time implementation of ARTC, based on these data, ARTC anticipates conflicts by simulating forward in time to detect potential deviations from the timetable or mutually incompatible requests by different trains for the same resources. If a conflict is anticipated, ARTC then triggers ADP algorithm to generate a set of feasible actions, with the help of a feasibility check module and the microscopic simulator, and evaluates each of the possible actions by performing explicit short-term delay (cost) evaluation and long-term delay approximation with the help of reinforcement learning (RL). The ADP then chooses the best performing action from the feasible action space, which is then reported as the optimised traffic control plan.

If Newton-based LSTD is deployed, the RL module needs to be trained before implementation can begin (this is not the case for Newton-based TD learning). This training can be done off-line, using historical data and the simulator, to arrive at robust RL parameters using methods discussed in the previous Chapter. An RL update module will carry on improving and updating RL parameters during real-time implementation.

4.2.2 Microscopic traffic simulation

Railway traffic simulation is a viable, efficient and accurate method of evaluating train movement in small railway networks. Simulation has been used for railway traffic research

[111, 112, 24, 113] and for industrial use, with commercial products such as RailSys [114] used for various purposes such as timetabling.

ARTC requires detailed microscopic traffic simulation to evaluate train movements inside the control area, therefore, a railway traffic simulator has been developed and implemented for this study in the Python programming language. The purpose of this is to facilitate explicit evaluation of cost g in the short-term in equation (4.4). Development of this microscopic simulator is to address three points:

i) Devise all feasible sequence permutations for the junction, to avoid resource incompatibility arising from unacceptable train plans,

ii) To calculate the short-term traffic state of the control area under consideration, including train entry and exit times for individual block sections, and,

iii) To evaluate performance of the ADP framework under different delay scenarios.

This simulator employs Lomonossoff's equation [115] to represent train movements inside each block section for the specified infrastructure by taking into consideration all the dynamic and static data described earlier in this Section.

Train movement is influenced by its tractive force, its mass and the resistance forces due to external factors such as air resistance [116]. Lomonossoff's equation is based on Newton's second law and is given as

$$\rho m a_t \;=\; F_{tr}(t) - F_b(t) - F_R(v_t) + F_{grad}, \tag{4.6}$$

where ρ is the rotating mass correction factor (set to 1.04 in this thesis), m is mass of the train, a_t is acceleration at time t, $F_{tr}(t)$ is the tractive force, $F_b(t)$ is the retardation force, $F_R(v_t)$ is the resistance force and is dependent on instantaneous speed v_t according to Davis' equation

$$F_R(v_t) \;=\; A + Bv_t + Cv_t^2, \tag{4.7}$$

where A, B and C are constants, dependent on the control state (*e.g.* motoring), and F_{grad} is the force due to gradient, defined as

$$F_{grad} = m_{total}\, a_{gravity}\, sin(\xi), \tag{4.8}$$

where m_{total} is the tare mass plus passenger mass, $a_{gravity}$ is acceleration due to gravity, and ξ is the slope angle. A train travelling uphill experiences negative F_{grad} and positive when travelling downhill. Furthermore, $F_R(v_t)$ can be derived from a look-up table based on current speed v_t, instead of calculated using Davis' equation (4.7).

Using data specified in the previous Section and train movement relationships presented in this Section, microscopic simulation of railway traffic can be utilised to evaluate different traffic scenarios in detail. The microscopic simulator developed in this thesis simulates train movements by taking into account all safety constraints imposed by conventional two, three and four aspect railway signalling systems by using train characteristics (*e.g.* available tractive force), signalling system (eg. block lengths and signal aspects), interlocking constraints, and station stops.

At the heart of this microscopic traffic simulator is the calculation of continuous braking curves, which are calculated to stop trains at the end of each block section and station, and for transitions into lower speed limits. Braking curves calculated for scheduled station stops and entry into lower speed limit zones are always respected, though ones to stop trains at the end of blocks are only enforced if the next block is occupied as indicated by the signalling system. Thus, all safety and operational requirements are respected, and infrastructure data and the movement of trains is represented in detail. The temporal resolution adopted here for this simulator is 0.1 second.

4.3 Realisation of the adaptive framework

Subject to the formulations and assumptions presented in this Chapter, the adaptive railway traffic control using ADP with integrated reinforcement learning (RL) is summarised in

Algorithm 11. In this algorithm, the embedded simulator is used to determine the complete feasible solution space at each stage by computing all sequence permutations, and ignoring sequences that cannot be realised in practice, *e.g.* on plain track a leading train from a stream cannot be overtaken by a following train. The complete feasible solution space is then used by the ADP framework to generate optimised solutions, where the simulator is again used to simulate forward the optimiser's decisions and check whether these can be realised according to the signaling system and other railway operational constraints.

The ADP framework computes entry sequence decision u_t^* at the conflict point, and by simulating interactions between all trains based on the decisions, the simulator computes exit times from the control area which are used to calculate individual train delays and hence the one-step cost $g(.)$ for steps 1 to T. Therefore, the optimiser is informed of the short-term performance beyond the conflict point. The long-term approximation $\widehat{J}(.)$ is then used to estimate trains performance (*i.e.* total consecutive delay of all trains beyond T) also at the exit from the control area for trains that are planned to enter the system in the future. Thus, decisions are made from a combination of short-term and long-term performance estimates, one explicitly calculated and the other approximated.

Recall that in this thesis, it is supposed that sufficient real-time data are available for railway systems to anticipate future arrival times into the control area, and other data are available that describe the probability of events occurring in operations, *e.g.* distributions for dwell times at stations. Furthermore, it is supposed that the latency of the system to receive data, calculate plans and deliver optimised sequences to the signaling system is short, *i.e.* the communication and computation delays are small so that control decisions can be calculated and implemented promptly.

The ADP procedure adopted can therefore be summarised as follow: the movement of the next T trains is represented explicitly, features ϕ based on the train times in the control area are extracted, and the optimal decision u_t^* are then calculated using (4.4). This requires evaluation of performance for u_t^*, which is then used to update approximation function by comparing the approximation $\widehat{J}(s_t, r_t)$ to the optimised performance $\widetilde{J}(.)$. The framework

then implements the first part of the calculated plan, *i.e.* u_1^* or the first train to be given movement authority into the junction, and consider the next T trains, *i.e.* $t+1$ to $T+1$. The framework then iterates through all remaining trains to produce the complete sequence of trains.

Step 1. Initialisation:

Choose an initial state s_0, Initialise parameter vector r_0,

Choose the horizon T stages to be calculated explicitly, Set $t = 0$,

Initiate learning rate η_0 (where required).

while *trains remain in the control area awaiting decisions:* **do**

Step 2. Receive current traffic state s:

2.1 Set $t = t+1$,

2.2 Extract features $\phi(s_t)$ from state s_t.

Step 3. Evaluate control decisions u:

3.1 Find the optimal sequence of decisions π_t^* using equation (4.4).

Step 4. Update approximation function $\widetilde{J}(.)$:

4.1 Calculate new observation $\widetilde{J}(s_t)$ using equation (4.3),

4.2 Calculate current approximation $\widehat{J}(s_t, r_t)$ using equation (3.10),

4.3 Calculate the T stage temporal difference δ_t at the current stage t using equation (4.5),

4.4 TD learning: Update parameter vector r_t to $r_{t+1} = r_t + \eta_t \Delta r_t$, where Δr_t is calculated according to the TD learning strategy adopted *i.e.* Algorithm 9 or 10.

Step 5. Implement decision:

5.1 Implement π_1^*, and update the state from s_t to $s_{t+1}(s_t, u_t)$,

5.2 Return to step 2.

end

Algorithm 11: Adaptive railway traffic control using ADP.

Initial values for the RL parameters r of equation (3.10) can be estimated off-line during a period of training using a combination of simulation and historical data. In real-time (on-line) implementation of Algorithm 11, a conflict anticipation module can be added where the microscopic simulator simulates forward in time using reliable estimated times of arrivals (ETAs) into the control area. This means ARTC will only be utilised if there are deviations from the timetable, otherwise run as is planned in the timetable. Notwithstanding this, producing plans in cases where all trains are running as scheduled may help with better RL parameters, helping to achieve better approximations $\widehat{J}(s_t, r_t)$ of long-term performance.

The framework presented here therefore uses data generated by trains to learn the underlying process of railway traffic in real-time to optimise signalling decisions. Few in the growing and rich literature available on RTM consider data-based approaches to solve this complex optimisation. As such, the contribution of this thesis to railway traffic management is therefore presented in Algorithm 11 (ADP adopted to work for RTM), Algorithm 9 and 10 (learning approaches that are different to RL literature and have been developed and tested for RTM).

4.4 Summary and discussion

As trains deviate from their scheduled timetable, they affect other services also following trains at points where the right of way is not exclusive to one direction of traffic, and minor deviations can magnify and propagate through the railway network. This can result in poor performance. DP can be used to model this as a sequential optimisation problem, though, solving Bellman's equation to derive optimised control plans in real-time is not practical. By using ADP to replace the optimal value function with an approximation, the computational effort required to solve the problem becomes appropriate from a practical point of view.

To solve RTM using ADP, the state-space of the problem is represented as stages, charac-

terised by the number of trains remaining in the control area awaiting sequence assignment, and the passage of one train through the conflict point is considered as transition from one stage to the next. In this formulation, by breaking down each decision (assignment of right-of-way to one train over others) into stages, ADP focuses computational effort on explicit evaluation of the next T trains where the associated real-time data is more reliable and decisions imminent and replaces the long term performance with an approximation. This approximation is then improved as time goes on using reinforcement learning (RL) techniques discussed in the previous Chapter.

Assuming the objective function is defined as a delay estimate, then the T-step explicit performance evaluation can be calculated using a purpose-specific microscopic simulation of the short term. This can be used to estimate delays to the next T trains awaiting right-of-way assignment under each feasible decision, and the long term evaluation is approximated using $\widehat{J}$ to represent delays of all other trains after the T^{th} train under optimised control following on from the state of the system after train T is served using the decision under consideration.

To help depict how ADP can be adopted for RTM, the sequencing of trains was used to explain how this process can be formulated. This does not mean that the adaptive controller developed in this Chapter can only be used for sequencing of trains at junctions. Similar formulation can be used to capture the characteristics of the problem for considerations such as re-routing and/or re-platforming. Given an appropriate choice of feature vector for the learning algorithms that effectively characterises the state and at the same time distinguishes among the decisions, the fundamentals of the framework presented here will stand. This would require the framework to evaluate explicitly the one-step cost g for a decision-space that includes re-routing and/or re-platforming of trains. This system can then be implemented either as a decision support tool to signallers or as a fully automated TMS.

ARTC relies on good quality real-time data from the railway network to optimise scheduling decisions. It is expected this will be provided by the envisaged ETCS-type signalling

systems. Real-time data from these digital systems' positioning sensors could be communicated via train-borne communication systems to provide good quality data in a short period of time and integrated with TMS. The commitment to implement ETCS-type signalling systems in Europe and similar systems around the world provides good opportunities for successful implementation of TMS in practice as the majority of the physical equipment required to provide real-time data to TMS algorithms are planned to be installed on many railways around the world. Systems such as ARTC can exploit the data transmitted for signalling purposes and provide re-scheduled plans to signallers or directly to the ETCS system.

High quality railway traffic simulators are important for effective TMS. As well as anticipating conflicts on the railway network they can be used to derive the feasible decision-space from the current state of traffic. Additionally for ARTC, they are employed to evaluate the short-term cost of feasible decisions and also used to train the RL component of ADP. It is likely that with future increases in processing power, simulations can evaluate large and complex railway networks accurately and in timely manners, therefore the methodology presented in this thesis is likely to become even more computationally efficient in the future.

The main distinguishing factor of the present ADP framework is its ability to exploit real-time data to facilitate optimised asset operations on railway networks. The framework presented in this Chapter uses data generated by trains to learn the processes underlying railway traffic in real-time to optimise signalling decisions. It does not guarantee optimality of solutions and trades off practicality over optimality. Little of the literature available on RTM considers data-based approaches to solve this complex optimisation.

Therefore, the present ADP framework is a systematic and self-tuning approach which considers the consequence of control decisions downstream of the conflict point in a timely manner and is suited for practical implementation. It automatically adapts parameters r to the objective function and the prevailing traffic on the rail network, allowing for better usage of real-time data generated by trains.

Chapter 5

Numerical Experiments

"Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things." - Isaac Newton

To evaluate our adaptive railway traffic control (ARTC) approach using the Newton-based TD learning techniques, infrastructure and operational data for a section of Great Britain's railway was used to conduct numerical experiments. In this Chapter, we first introduce the case study network in Section 5.1 and discuss challenges and significance of the considered network in terms of mainline operations. Then, we set out our method for perturbing the timetable and present the environment in which our experiments are conducted. Finally in Sections 5.2 to 5.5, our results of employing ADP for railway traffic control are presented and discussed. In the remainder of this Chapter we refer to our ADP approaches solely according to their learning techniques *i.e.* one-step TD and LSTD.

5.1 Case study

Part of the East Coast Main Line (ECML) in the UK is used as an example that represents a high capacity main-line. The section of network used for our case study is located on the southern part of the ECML between Stevenage and London, runs for approximately 70 kilometres, and includes 32 stations. Figure 5.1 is a schematic of the considered network which is electrified along its whole length at 25 kV AC and runs on conventional 4-aspect

signalling.

Two points have been identified on this network which are critical in terms of traffic flow.

- Digswell junction: The area in the vicinity of Welwyn North station;
- Finsbury Park area: The area in the vicinity of Alexandra Palace station leading towards a major station: Finsbury Park.

These are junctions where services with different rolling stock characteristics and stopping patterns interact. Such junctions are known to present major risks to the maximal capacity utilisation of networks and at the same time present the most significant opportunities for effective regulation of traffic.

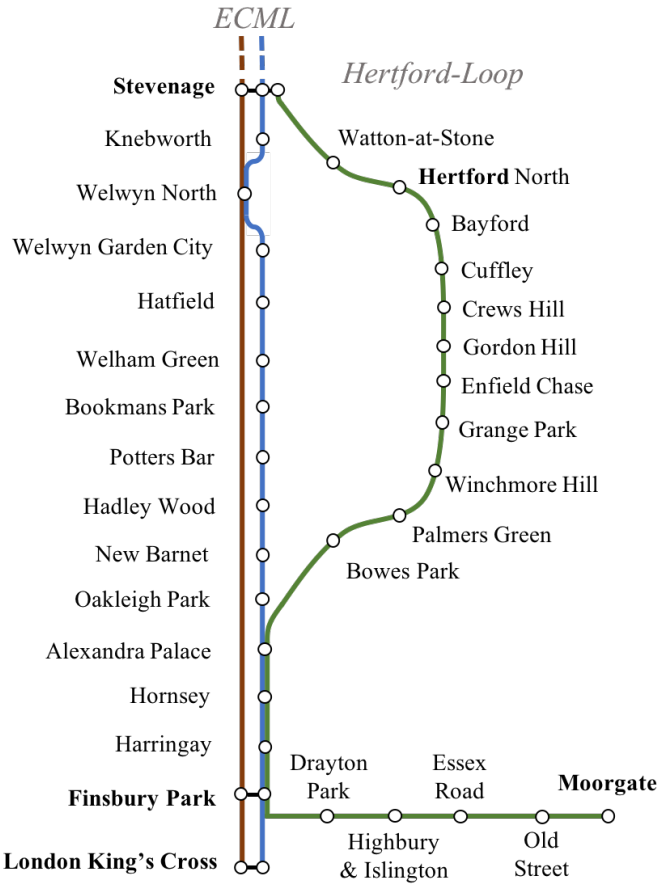


Figure 5.1: Schematic of the southbound, southern section of the East Coast Main Line. Different colours of track present particular services, with varying train mechanical capabilities, stopping patterns and train operators.

5.1.1 Challenges and significance of ECML

The ECML consists of four tracks, one fast and one slow in each direction, for most of its length. The infrastructure around the village of Digswell is an exception where four tracks narrow to two tracks over the Welwyn Viaduct, located between Welwyn North and Welwyn Garden City stations, which carries trains over River Mimram, and through two tunnels north of Welwyn North station. It is on this part of the network that High-Speed Intercity services (ECML HST) and commuter services (ECML commuter) with frequent stops must negotiate usage of the same tracks through Welwyn Viaduct. The problem of conflicting requests for usage of tracks is exacerbated by commuter trains with planned stops at Welwyn North station, which block the line while dwelling at this station.

Welwyn Viaduct is an historic structure that was opened in the mid-19th century and therefore by law cannot be altered or demolished, as it is a Grade II listed building. This makes replacement of this known bottleneck expensive, time consuming and practically impossible for the infrastructure manager. It is therefore a prime target for implementation of such tools as envisaged in this thesis, and so is ideal for our numerical experiments.

The Hertford Line or Hertford-Loop, is a branch of the ECML where commuter trains run between Stevenage and Moorgate stations, stopping at local stations such as Hertford North among others to/from London. The line is around 40 km long, and rejoins the ECML north of Alexandra Palace station. This is where frequently stopping Hertford-Loop commuter trains interact with ECML commuter trains.

Efficient management of railway traffic in these two areas is of paramount importance to the ECML, as they represent two of the major critical points on the rail link connecting the eastern side of Great Britain, from London and the South East of England to Scotland.

In the first instance, we only consider controlling the traffic at Digswell junction in Sections 5.2 to 5.4 to conduct preliminary tests using the ADP framework and also test ADP's effectiveness in stochastic railway environment; then, we consider distributed deployment of ADP at Digswell junction and in the Finsbury Park area in Section 5.5.

In this thesis, we simulate the 2018 timetable for a weekday between 7:00 am and 10:45 am.

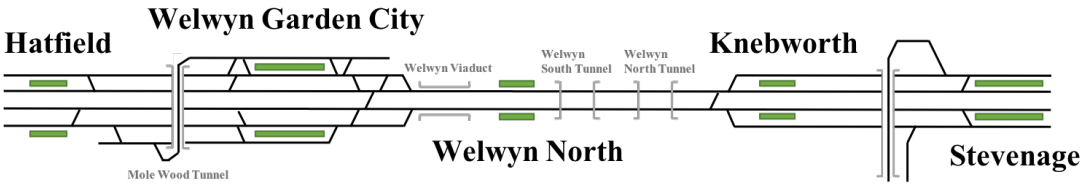


Figure 5.2: Infrastructure layout of the Digswell junction and the surrounding stations.

Northbound and southbound services do not interact in our timetable, so can be controlled independently.

Therefore, for experiments on Digswell junction only (Sections 5.2 to 5.4), we focus on sequencing trains at the bottleneck in Figure 5.2 for southbound services which includes, 25 ECML HST and 6 ECML commuter services with frequent stops, that run from Stevenage towards Hatfield.

For the distributed deployment experiments of Section 5.5, we focus on southbound services from Stevenage station into London, in total 42 services. This consists of 25 ECML HST services and 6 ECML commuter services with frequent stops on the mainline of ECML and 11 Hertford-Loop commuter services on the Hertford-Loop and into London. Hence, in all experiments the morning peak period on ECML towards London is considered.

5.1.2 Perturbing the timetable

We perturb our timetable by delaying all trains that dwell at Stevenage Station. This includes all ECML and Hertford-Loop commuter services, as well as some ECML HST services. More than half of the trains are perturbed by sampling from a Weibull distribution with shape parameter 1.8 and scale parameter 8, producing mean delays of 7.1 seconds (Figure 5.3 [52]). The individual delays can then be scaled with random factors to generate different traffic scenarios. As for the trains that do not stop at Stevenage, some may be delayed due to delayed services dwelling at Stevenage.

The railway operating environment has substantial stochastic elements, so to assess our framework, in Sections 5.3 and 5.4 using the Digswell junction, we evaluate each traffic

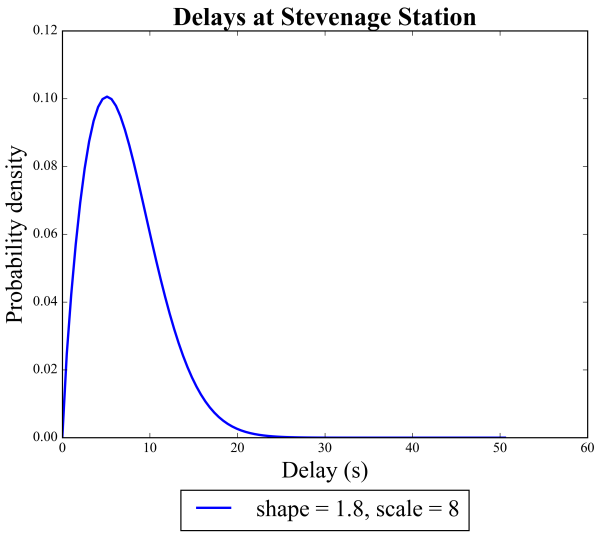
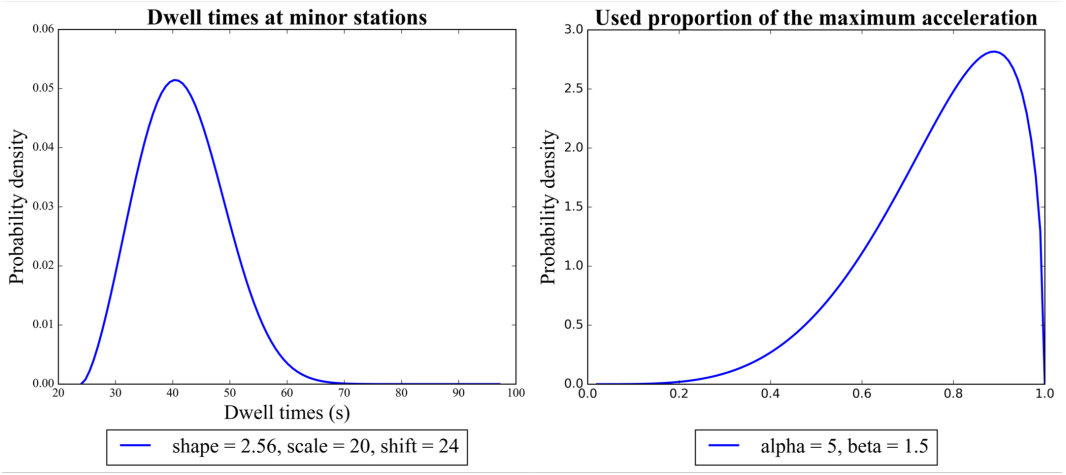


Figure 5.3: Probability distributions of delays at Stevenage Station without scale factors.

scenario in stochastic environment with Monte-Carlo simulation with 30 combinations of i) dwell times per station, and ii) running times per block section that, are generated by random draws from associated probability distributions. In the absence of empirical dwell time and sectional running time distributions for our test case, we have assumed the following:

- Station delays: The Digswell case study network includes five stations; of these, all except Stevenage Station are minor stations at which a relatively small number of services have planned stops. The dwell times for minor stations are generated from a Weibull distribution with 2.56, 20, 24 as shape, scale and shift parameters respectively. This distribution is reported by Quaglietta et al. [52] for dwell times at minor stations in the Netherlands and is presented in Figure 5.4.a . The minimum dwell time at a minor station on ECML test case network is 30 seconds. If a sample returns a dwell time less than the minimum dwell time, then 30 seconds is used; taking this into account, our mean dwell time at a minor station is 41.8 seconds.

- Section running time delays: Train drivers vary in their driving behaviour. Factors affecting this include their route knowledge and the present weather conditions. Although this variation in behaviour may affect short term predictions of traffic state, it is difficult to find a unified probability distribution that represents this. In our study,



(a) (b)

Figure 5.4: Probability distributions of (a) dwell times at minor stations, and (b) the proportion of the maximum acceleration used by a driver on a block section.

we use a *Beta* distribution with shape parameters α and β as 5 and 1.5 respectively, shown in Figure 5.4.b , to represent the proportion of the maximum train acceleration that is used by a driver in a block section. Similar to delays at minor stations, we set a minimum acceptable proportion as 0.65; therefore, the mean proportion of maximum acceleration used on a single block section in our study is 0.79. The purpose of this is to introduce additional stochasticity and so to test the effectiveness of this ADP formulation.

In this study, the First-Come-First-Served (FCFS) heuristic provides a baseline for comparison of performance. No estimate for the globally optimal performance is considered because the computational complexity of RTM makes this impractical. All of the control methods were evaluated using realised dwell times and sectional running times, whilst the ADP approaches did not use future dwell and running times to calculate control decisions.

5.1.3 ADP control parameters

An issue in using one-step TD learning is the choice of moderating step size η in updating the parameters r_t . A common approach in stochastic approximation is to use a stage-varying rate: we adopted constant value for η of 0.1 and 1 in the experiments of this

Table 5.1: Average consecutive delays (s) by various discount rates.

θ	0.05	0.1	0.11	0.12	0.13	0.14
One-step TD	14.425	14.434	14.437	14.434	14.435	14.420
LSTD	14.441	14.433	14.442	14.446	14.445	14.449
θ	0.15	0.16	0.17	0.18	0.19	0.2
One-step TD	14.417	14.443	14.435	14.438	14.453	14.443
LSTD	14.418	14.417	14.438	14.432	14.435	14.439

Chapter, so corresponding to the exponentially weighted variant. The discount factors α for use in equations (3.7) and (3.8) were calculated as $\alpha_t = e^{(-t\theta)}$. Table 5.1 presents performance of various discount rates. It can be seen that θ at 0.15 results in good performance for both one-step TD and LSTD, and therefore in this Chapter all experiments are conducted with the discount rate set at $\theta = 0.15$ (*i.e.* a discount of 15% for each successive train).

The computational time for TD and LSTD are similar. This depends on the choice of T in equation (4.4) to compute the explicit part of the cost function $g(.)$. Computation of $g(.)$ was found to be the most burdensome part of the present ADP framework, where we evaluate train movements individually. This choice is delicate, as larger values of T result in better performance, however they come at the cost of computational efficiency especially in the case of implementation in stochastic environment; this is presented in Table 5.2 for stochastc and deterministic implementations. In the remainder of this thesis, for experiments in stochastic environment we set $T = 3$, and for experiments in deterministic

Table 5.2: Average consecutive delays (s) and computational time of various choices T.

T	One-step TD	LSTD	Time per stochastic / deterministic decision (s)
2	15.80	15.79	0.73 / 0.06
3	13.87	13.87	1.10 / 0.11
4	13.77	13.76	2.86 / 0.23
5	13.73	13.73	6.03 / 0.44
6	13.75	13.75	11.78 / 0.74
7	13.75	13.73	20.63 / 1.47

environment we set $T = 4$.

All numerical experiments were conducted by computer simulation, implemented in Python 3.4 running under Windows 10 on a PC configured with Intel Core i7- 4790 CPU and 32 GB RAM.

5.2 Objective functions for ARTC

We considered three objective functions to evaluate the cost during a single stage g in equation (4.4) based on conversations with railway operations professionals: 1) Minimising total consecutive delays, 2) minimising total delays, and 3) minimising total running times inside the control area:

$$\text{Objective 1} = \sum_t [C_t]_+ , \tag{5.1}$$

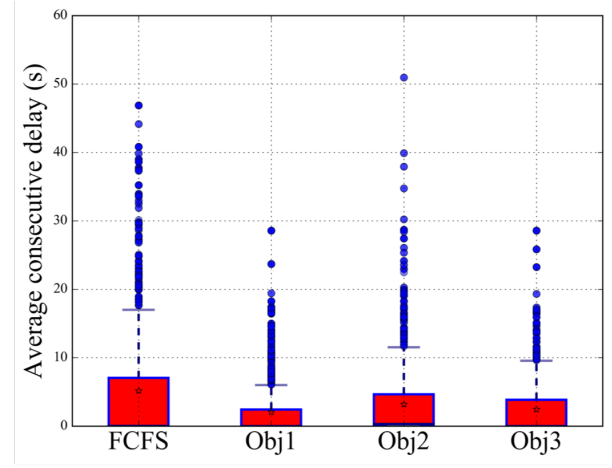
$$\text{Objective 2} = \sum_t L_t^e + \sum_t C_t , \tag{5.2}$$

$$\text{Objective 3} = \sum_t R_t + \sum_t [C_t]_+ , \tag{5.3}$$

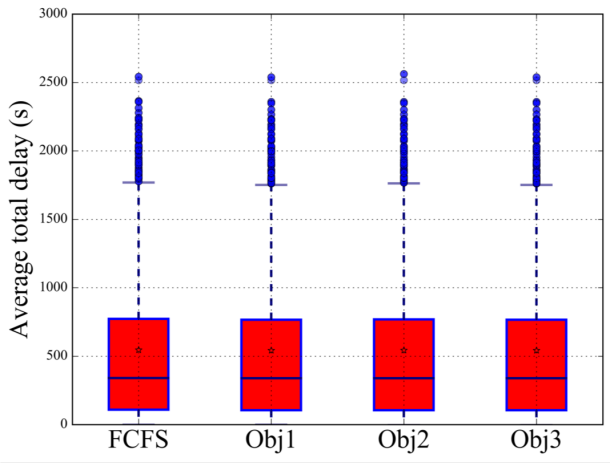
where C_t is the accumulated consecutive delay inside the control area per stage, L_t^e is delay on entry to the control area, and R_t is the scheduled running time inside the control area. These objective functions were chosen as they are not too complex as to divert the focus of this study and at the same time important enough to be considered reasonable choices.

For the Digswell control area of Figure 5.2 and simulating deterministically, we randomly generated 60,000 individual train delays (2000 morning peaks) using the distibution of Figure 5.3 [52] resulting in a mean entry delay of 9 minutes 56 seconds at Stevenage Station.

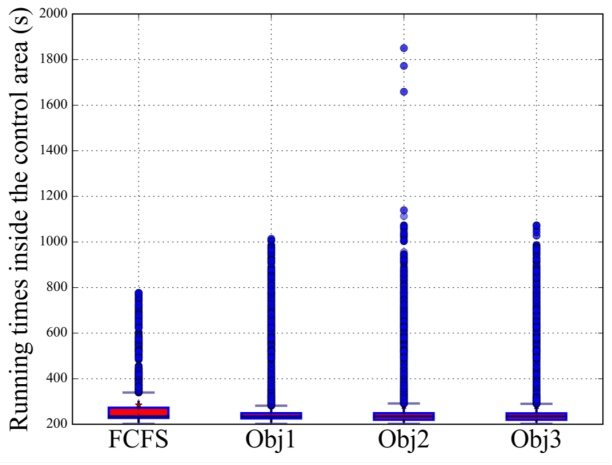
Result indicates that the ADP approach achieves superior performance compared to FCFS using all objective functions, though performance varies among the objective functions. Table 5.3 presents comparison of performance based on delays on the exit of the Digswell control area for each of the objective functions using LSTD learning, and hence provides a combined view of performance.



(a)



(b)



(c)

Figure 5.5: Average (a) consecutive delay, (b) total delay, and (c) train running times inside the control area, per scenario comparing performance of ADP with LSTD learning under different objective functions.

It is apparent that Objective 2 does not perform well: the performance under ADP with this objective is worse than with the others, even in terms of its own measure. Noticeably, Objective 1 and Objective 3 both perform better in terms of Objective 2's primary objective *i.e.* minimising total delays.

Figure 5.5.a presents mean consecutive delays for each simulated morning peak (each delay scenarios). It is evident from this graph that Objective 1 performs the best in all percentiles shown on the graph. Objective 2 has more variation after the 90^{th} percentile compared to others. The maximum for Objective 2 is higher than the maximum for FCFS. Figure 5.5.b presents mean delays for each delay scenarios. Improvements in total delays are small compared to FCFS. The reason for this is the limited opportunity for bringing trains back to their planned schedules; all objective functions improve total delays with Objective 3 performing the best. Figure 5.5.c presents train running times for all trains. All objectives perform favourably compared to FCFS in this measure up to the 90^{th} percentile. Higher maximums in Figure 5.5.c for our objectives indicate trains that were held at the junction to allow other trains through. This action is beneficial as is evident from the improved overall performance.

Considering findings of Figure 5.5, it is evident that performance of the ADP approach is sensitive to the choice of objective function and unsuitable choice can result in poor performance. It is shown that, in the case of railway rescheduling, an objective function may be outperformed by others in terms of its primary objective. In the remainder of this thesis, to evaluate ADP in detail, we use Objective 1 in Sections 5.3 and 5.4, and Objective 3 in Section 5.5.

5.3 Feature selection for ARTC

There are numerous possibilities for the features ϕ in equation (3.10) that are extracted from the state s_t at each stage t . Although many features that represent the state and can be extracted could be used for approximation of the value function $J(.)$, not all combinations will

Table 5.3: Comparison of objective functions against all performance measures.

	Mean consecutive delay (s) - Obj.1	Mean total delay (s) - Obj.2	Mean running time (s) - Obj.3
FCFS	5.17	547.00	285.11
ADP-Obj.1	2.08	543.04	281.06
ADP-Obj.2	3.21	543.48	281.51
ADP-Obj.3	2.44	542.60	280.61

necessarily benefit performance. The choice is important as it may affect both effectiveness and computational efficiency of the ADP frameworks.

To investigate effects of this choice of features ϕ on performance, stochastic traffic scenarios were simulated for Digswell control area with mean initial delay for all trains, measured just downstream of Stevenage Station, as 15 minutes 2 seconds. Combinations of features were then tested for LSTD learning under identical conditions to compare performance. We extracted and tested all 15 combinations of the four features:

a) Remaining scheduled running time of trains in the control area,

b) Train delays as measured just after Stevenage Station,

c) The time difference between services at the conflict point according to the current plan, and

d) The service headways according to the current plan.

Figure 5.6 presents performance of all fifteen combinations of these features together with a controller that used none. The measure of performance is the mean consecutive delay of all trains when they reach Hatfield Station. For reference, the FCFS performance for the traffic scenario was 20.48 seconds (not shown on the graph because it is substantially greater than all of the LSTD values). Most combinations of features perform similarly to or worse than none at all that corresponds to a T step greedy heuristic without any approximation function, *i.e.* $\widehat{J} = 0$, however, the three combinations {a, c}, {a, b, c} and {a, b, d} did improve

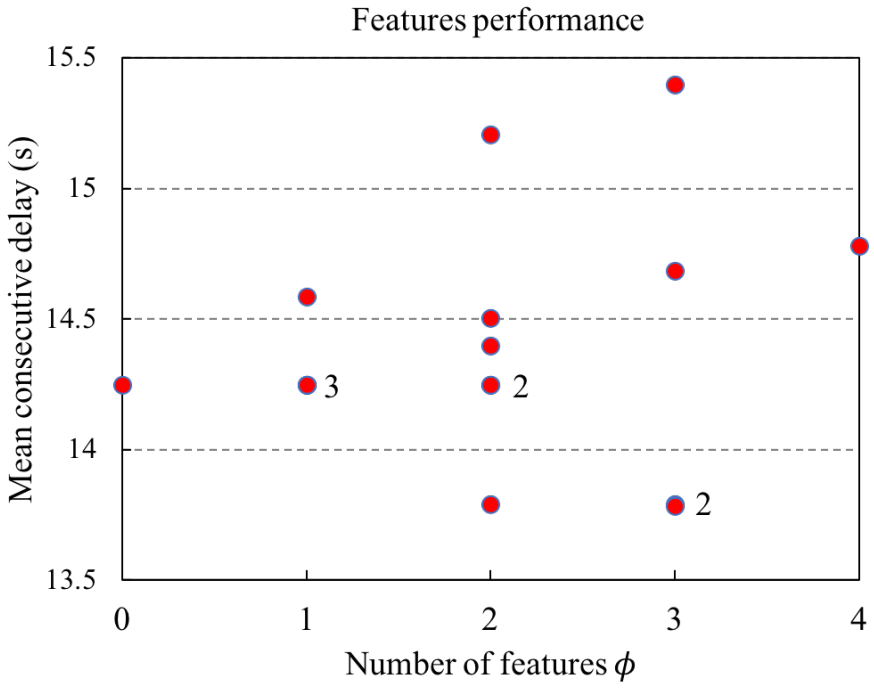


Figure 5.6: Performance of various feature combinations (multiple coincident values are indicated by number).

performance. Of these, the combination {a, b, d} produced the absolute best performance with a mean of 13.78 seconds per train followed closely by combinations {a, c} and {a, b, c} that both produced means of 13.79 seconds per train compared with the 14.25 seconds per train for the controller that used no features. The combination {a, c} is adopted for use because it performs nearly as well as the best combination but uses one fewer feature. The worst performing combination was {b, c, d} with 15.4 seconds per train. In all combinations that included both c and d, poor performance was observed.

More combinations of features resulted in worse performance than using none at all, which shows the importance of appropriate feature selection for ADP with RL. In the remainder of this Chapter, we investigate in detail the performance of the ADP frameworks using the {a, c} features, therefore we extract scheduled running time of remaining trains in the control area ($\phi_1 = a$) and the time difference between services at the conflict point according to the current plan ($\phi_2 = c$) for the approximate value function $\widehat{J}(.)$.

5.4 ARTC and the stochastic railway environment

To evaluate ADP's performance under uncertainty, two traffic scenarios were considered for Digswell control area using Objective 1 (*i.e.* minimising total consecutive delays). For each, in total 30,000 random train delays were generated from the Weibull distribution of Figure 5.3 [52] for Stevenage Station, equating to 2,000 morning peaks. The delays are then scaled by random factors to create mean initial delay, for all trains as measured just after Stevenage Station, as 4 minutes 37 seconds for *Traffic Scenario A*, and 14 minutes 58 seconds for *Traffic Scenario B*. To put these into perspective, mean timetable headway between all services on the shared section of Figure 5.2 is 4 minutes and 39 seconds. Accordingly, we aim to investigate ADP performance during moderate perturbations with traffic scenario A, and instances with more substantial perturbations using traffic scenario B.

In this Section we adopt all of the adjustment Δr for LSTD (*i.e.* $\eta_t = 1$) and, to compare traffic scenarios A and B we only adopt a tenth of the adjustment Δr for TD (*i.e.* $\eta_t = 0.1$). Towards the end of this Section, adopting all adjustments Δr is also considered for TD.

5.4.1 Performance analysis in stochastic environment

Table 5.4 presents comparison of performance among FCFS, one-step TD and LSTD according to three measures of performance for each of traffic scenarios A and B. LSTD and one-step TD achieve almost the same performance, meaning the controls produced using the two learning methods are similar (*i.e.* they computed the same sequences in almost all cases). This need not mean they calculate the same approximations; indeed, the mean

Table 5.4: Comparison of performance in stochastic environment of Digswell control area for traffic scenarios A and B (seconds per train).

	Traffic Scenario A			Traffic Scenario B		
	FCFS	TD	LSTD	FCFS	TD	LSTD
Mean consecutive delay	19.35	14.0	14.0	21.33	15.21	15.22
Mean train running time	296.19	290.83	290.83	298.83	292.69	292.68
Mean total delay	295.21	289.85	289.85	918.93	912.80	912.79

Table 5.5: Percentiles of consecutive delays in stochastic environment of Digswell control area for traffic scenarios A and B (seconds per train).

	Traffic Scenario A			Traffic Scenario B		
Percentile	FCFS	TD	LSTD	FCFS	TD	LSTD
75^{th}	24.91	17.48	17.47	29.27	19.45	19.45
90^{th}	46.32	28.91	29.02	50.52	30.82	30.81

absolute error in approximation of value function $\widehat{J}$ in one-step TD was 63.8 for scenario A and 68.3 for scenario B, whereas for LSTD it was slightly greater at 64.5 for scenario A and 71.2 for scenario B. In both cases however, the approximations favoured similar control decisions. The ADP reduced mean consecutive delays by around 28%, mean running times by around 2% and mean total delays by around 1%.

As the primary measure for our objective function, mean consecutive delays reveal how one-step TD and LSTD contributed towards their principal goal. One interesting observation is the similarity of the mean consecutive delays in traffic scenarios A and B. It seems that initial delays do not influence strongly the consecutive delays incurred inside the control area; this can also be observed in Figure 5.7, which present boxplots for the mean consecutive delays.

The main improvements in the high values of delay in Figure 5.7 can be seen in the 75^{th} and the 90^{th} percentiles which are shown in Table 5.5. It is evident that the two learning techniques are similarly effective in reducing consecutive delays compared to FCFS whilst of the two, LSTD performs marginally better. This is also reinforced when a t-test was completed on the data concluding a t-value of 15.13 meaning a big difference between the results of TD / LSTD compared to FCFS.

As an indication of the rate of track occupancy within the test network, we study the running times of individual trains inside the control area. Figure 5.8 presents boxplots for train running times in traffic scenarios A and B. As with the consecutive delays, both TD and LSTD improve running times substantially in the 75^{th} percentile of Figure 5.8, as is shown in Table 5.6. Unlike the mean consecutive delays, there is a long tail in TD and

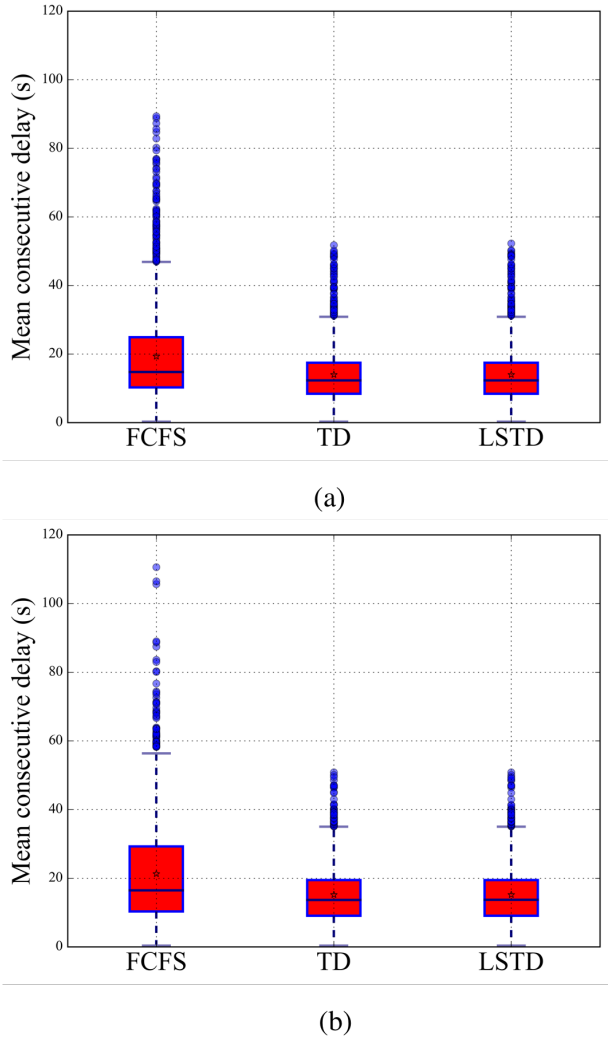
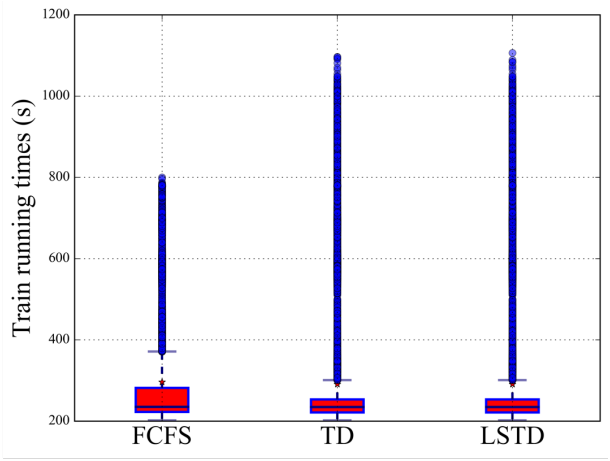


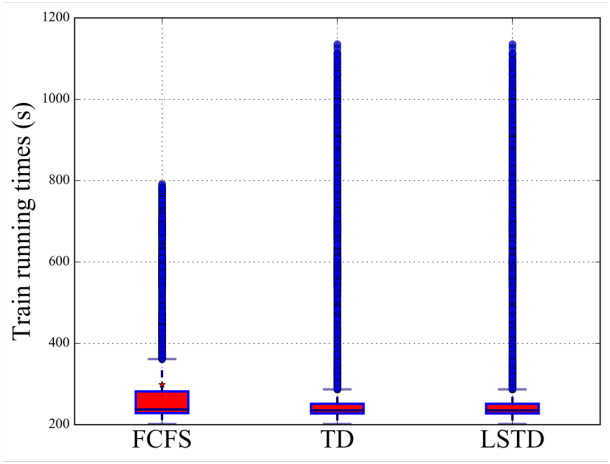
Figure 5.7: Boxplot of mean consecutive delay by scenario for (a) traffic scenario A, and (b) traffic scenario B.

LSTD individual train running times of Figure 5.8 with substantially greater high extremes. These arise from trains that are further delayed in the control area for the sake of gains in the principal objective of network performance. However, these actions may be especially unfavorable to certain train services so the objective could be developed further to achieve more equitable solutions.

Once more, in Figure 5.8, similar performance can be seen in the two traffic scenarios even though the initial delays differ. The similarities in consecutive delays and running times for both traffic scenarios suggests more attention needs to be given to train headways resulting from initial delays rather than the initial delays themselves. This can also be hypothesised



(a)



(b)

Figure 5.8: Boxplot of individual train running times inside the control area for (a) traffic scenario A, and (b) traffic scenario B.

Table 5.6: 75^{th} percentiles of train running times in stochastic environment of Digswell control area for traffic scenarios A and B (seconds per train).

Traffic Scenario A			Traffic Scenario B		
FCFS	TD	LSTD	FCFS	TD	LSTD
281.92	253.31	253.31	281.69	251.24	251.25

from performances in Figure 5.6 .

The improvements in total delays in Table 5.4 are relatively small. Because of the ways that railways operate, most of the initial delays may not be recoverable and indeed this is not included in the present objective. It should be noted that the improvements in mean consecutive delays, running times and total delays are similar in both traffic scenarios.

Table 5.7: Mean and standard deviation of parameters for TD with η_t of 0.1 and 1 for traffic scenario A.

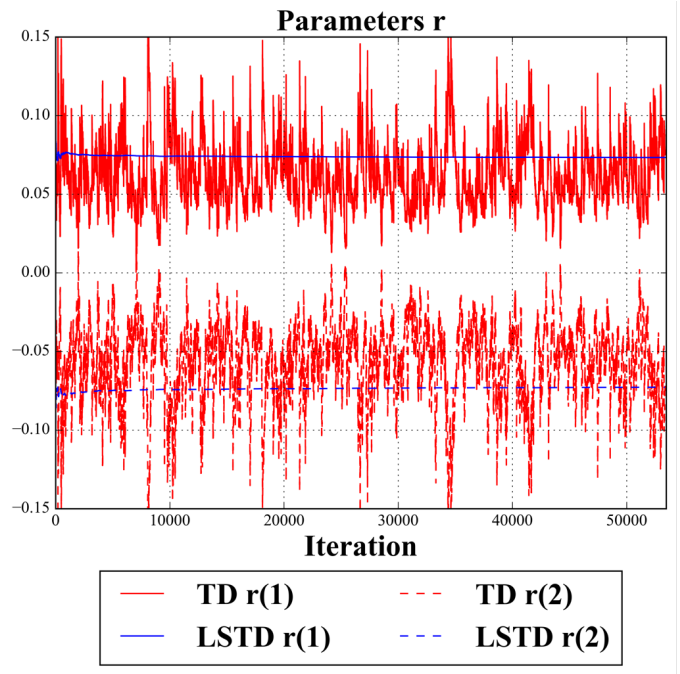
		Mean		Standard Deviation	
Parameters	η_t:	0.1	1	0.1	1
$r(1)$		0.063	0.072	0.021	0.060
$r(2)$		-0.057	-0.067	0.025	0.072

Comparisons of Figure 5.8 shows that in ADP, on occasion, a few trains are delayed substantially with the intention to achieve low average delay during the remainder of that run.

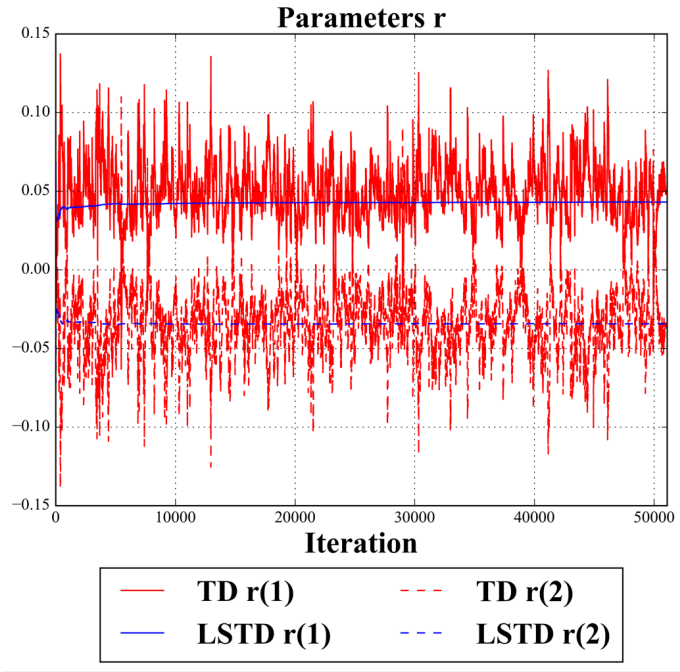
5.4.2 Comparison of learning methods

Figure 5.9 presents evolution of the parameters r under each of the TD and LSTD learning strategies in the two traffic scenarios. The parameters approach similar values notwithstanding the different mean entry delays. TD learning behaves similarly in both traffic scenarios, whilst LSTD parameters stabilise on larger values in scenario A compared to the more heavily delayed scenario B. Part of the reason for this behaviour may be that in traffic scenario A the time difference between services at the conflict point according to the current plan (ϕ_2) are smaller and more stable compared to traffic scenario B. In the presence of substantial perturbations to trains, greater values of ϕ_2 are extracted from the current state. Recalling that LSTD learning solves for parameters r by minimising the sum of squares of all temporal difference errors since the start of operation, it is perhaps not surprising that r values stabilise on smaller values for scenario B compared to scenario A. Also, for the same reason, LSTD becomes more stable as time goes on compared to TD learning which puts more emphasis on recent observation.

Depending on the values adopted for the modification parameters η_t , TD may be more adaptive than LSTD, and as shown in Figure 5.10, TD achieves smaller errors than does LSTD which suggest that TD tends to approximate the value function more accurately. The reason is that states are sequentially connected, so a good fit of $\widehat{J}$ at time t is likely to lead



(a)



(b)

Figure 5.9: Evolution and comparison of learning parameters for TD η_t of 0.1 and LSTD learnings for, (a) traffic scenario A, and (b) traffic scenario B.

to a good estimate at time $t+1$. Furthermore, one-step TD reacts to short-term changes in traffic on the network, whereas LSTD stays stable and approximates according to what has

been observed from the start of the operations.

Furthermore, Figure 5.10 shows that TD with $\eta_t = 1$ achieves even smaller errors as it adopts all adjustments to r after every observation which means it reacts to changes in traffic more strongly than it does with $\eta_t = 0.1$; this however comes at a cost of reduced

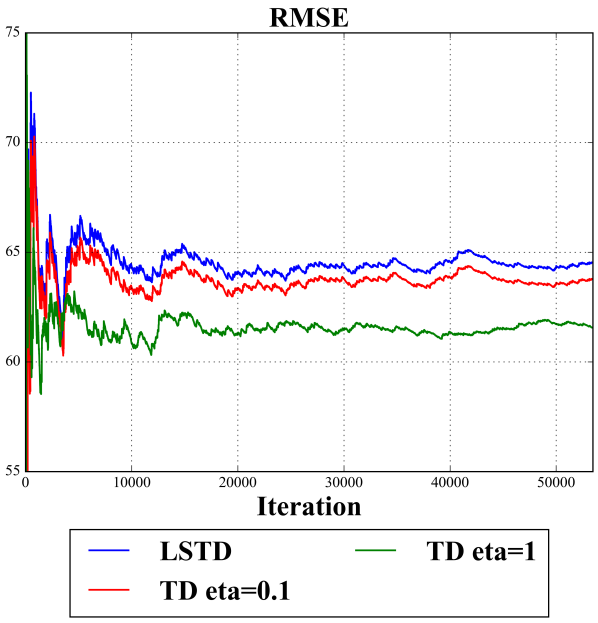


Figure 5.10: Root Mean Square Error of TD with η_t of 0.1 and 1, as well as LSTD learning for traffic scenario A.

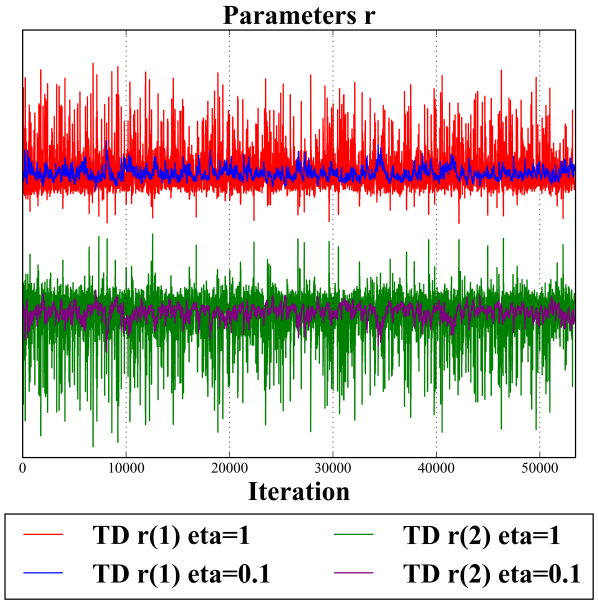


Figure 5.11: Evolution and comparison of learning parameters for TD with η_t of 0.1 and 1 for traffic scenario A.

stability in the parameters as is clear in Figure 5.11. It has to be noted that TD with $\eta_t = 0.1$ and $\eta_t = 1$ achieved similar performance. Because LSTD has no modification parameter to be tuned, the stabilisation in the parameters for LSTD is achieved automatically. There may be a better specification for the modification parameters η_t than a constant which could contribute to improved performance.

The results and the small difference between approximation and learning values presented in this Section show that both ADP approaches seem to perform well in railway traffic management in stochastic environment with similar potential for benefits in terms of operational performance.

5.5 Distributed ARTC

The aim of this Section is to investigate the effectiveness of ADP in tackling railway rescheduling for large and dynamic networks. Given that the goal of the RTM system is to achieve network optimisation in a timely manner, the ADP framework is deployed in a realistic simulation environment in a distributed setup across the southern section of the ECML, to control known critical areas of the network and evaluate network-wide performance.

5.5.1 Evaluation framework for distributed control

Having identified two areas as areas with significant opportunities for traffic optimisation, the framework is implemented to control the vicinity of: 1) Digswell, and 2) Finsbury areas (red shaded boxes A and B respectively in Figure 5.12), therefore representing a distributed control setup.

A microscopic railway traffic simulator called BRaVE (Birmingham Railway Virtual Environment), developed at the University of Birmingham, is used to simulate the movement of trains and the operations of the interlocking and signalling across ECML; this is different to the microscopic simulator embedded in the ADP framework. An *Application Programming Interface* (API) mechanism enables external software to control the scheduling of trains

Figure 5.12: Schematic of the study network, with red boxes indicating, A: Digswell control area, and B: Finsbury control area.

within the pre-defined control areas of the simulator. The API mechanism communicates *XML* format messages containing outgoing traffic state and incoming optimised sequence instructions for junctions in the specified control areas (*i.e.* Digswell and Finsbury control areas). These messages are relayed over a *UDP* server connection.

The ADP controller interfaces with the simulator via the UDP connection and communicates XML format messages. During a simulation run, traffic state messages are periodically sent to the ADP controller, which monitors the state of the network and optionally sends back optimised sequencing instructions; in this Section, traffic states are sent to the controller regularly every 5 minutes and also every time trains enter the control area. On receipt of these instructions that are returned to the simulator which implements them. This

Table 5.8: Mean running times (seconds) separated by control areas and service groupings.

	Mean running times in seconds			
	Digswell		Finsbury	
Service groups	FCFS	ADP	FCFS	ADP
All services	324.59	322.44	853.04	850.81
ECML HST	230.31	223.85	-	-
ECML Commuter	607.42	618.21	354.12	342.08
Hertford-Loop Commuter	-	-	1125.17	1128.3

process continues until the simulation ends at a pre-specified time.

We perturb the traffic, similar to the previous Section, by delaying all trains that dwell at Stevenage station. This includes all ECML and Hertford-Loop commuter services, as well as some ECML HST services. In total over 60% of trains are perturbed by an amount that is sampled from the Weibull distribution of Figure 5.3 [52]. These delays were then scaled with randomly selected factors ranging from 5 to 40. As for the trains that do not stop at Stevenage, some may be delayed due to following services that are delayed while at Stevenage. Once trains leave Stevenage, running times and subsequent dwelling periods are treated deterministically.

For the evaluation of this Section, Objective 3 of equation (5.3) is used, therefore, we focus on train running times inside the control area and on station delays as a measure of performance in the network as a whole.

5.5.2 Results for distributed control

In total 28,000 individual train delays were generated for Stevenage Station, equating to 1000 distinct morning peak periods, which produced a mean added delay of 2 minutes and 41 seconds on dwell times at Stevenage station. Therefore, the aim of this Section is to investigate effects of ARTC on normal operations across a large network.

Table 5.8 presents a comparison of performance between FCFS and ADP inside the control areas according to mean service running times. As well as overall performance, this table shows running times by service groupings of the timetable. It can be observed that

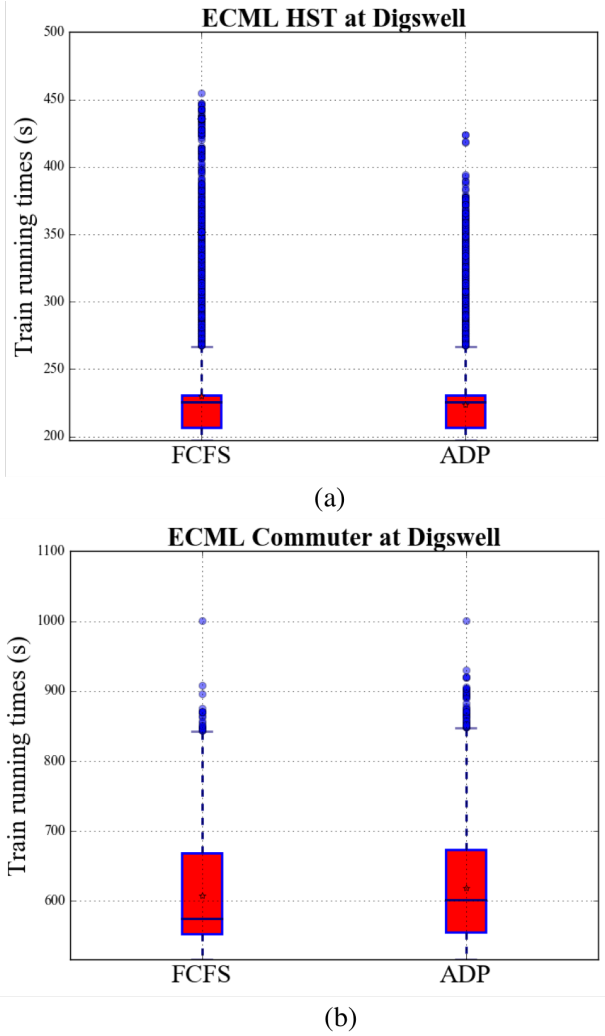


Figure 5.13: Running times of services inside Digswell control areas separated by service groupings; (a) ECML HST, and (b) ECML commuter services; star signs indicate mean values.

the mean running time of all trains has reduced, by 2.12 seconds for Digswell control area and 2.23 seconds for Finsbury control area. Given that the mean running time for Digswell and Finsbury, at full speed and with no conflicts, are 305.39 and 842.41 seconds respectively, the improvements represent 11% and 21% reduction in mean added running times for Digswell and Finsbury control areas, respectively. This directly equates to maximising track occupancies within the control areas compared to FCFS, and also reduction in consecutive delays, indirectly.

To further analyse the effects of optimisation on the performance of services, we disag-

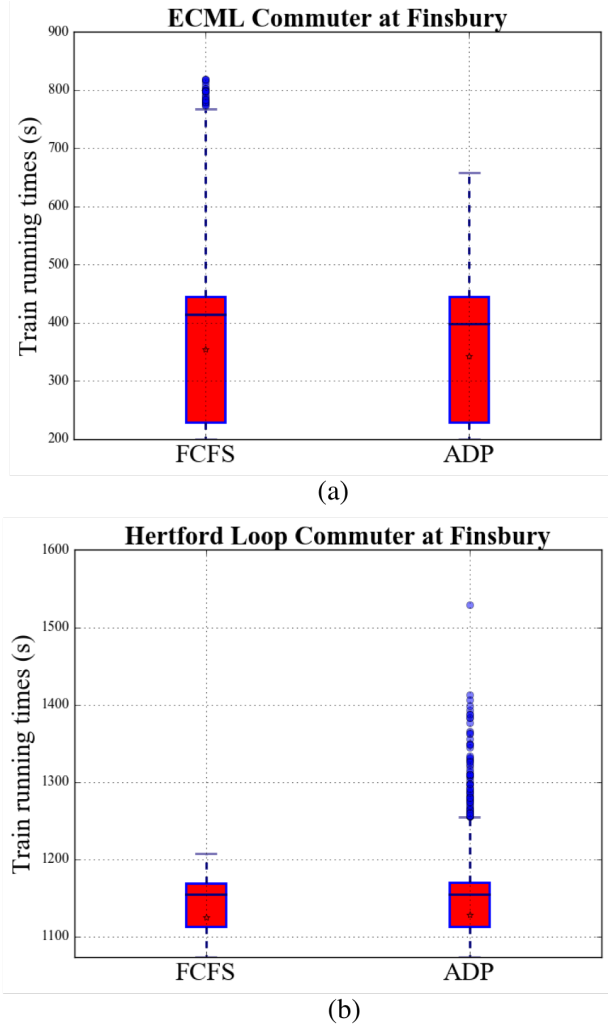


Figure 5.14: Running times of services inside Finsbury control areas separated by service groupings; (a) ECML commuter services, and (b) Hertford-Loop commuter services; star signs indicate mean values.

gregate the running times according to service groupings. When comparing running times inside Digswell control area in Figure 5.13.a and Figure 5.13.b and also in Table 5.8 , it is clear that ECML HST services benefit from the rescheduling, whereas ECML commuter services are disadvantaged. This trend is again repeated within Finsbury control area (Figures 5.14.a and 5.14.b) where ECML commuter trains benefit from optimisation and Hertford-Loop commuter trains do not.

By inspecting the sectional running times of the service groupings, it is evident that the beneficiary service groups require a shorter running time to traverse the control areas. As the control framework attempts to minimise the total running times of all trains, it is no surprise

Table 5.9: Like by like mean delays that exceed X seconds.

Major stations (N^o of trains)		Delays greater than (s): X: 0	60	180	300
London King's Cross (31)	FCFS	16.6	200.8	293.6	389.9
	ADP	15.1	180.0	248.7	287.5
Moorgate (11)	FCFS	6.3	157.1	311.0	466.2
	ADP	7.4	178.3	325.4	514.4
Finsbury Park (21)	FCFS	55.9	198.5	345.9	414.4
	ADP	55.1	194.8	312.4	343.7

that the overall effect of rescheduling when taking into account all trains, means that faster trains are favoured in the optimised control plans. In the case of Digswell control area, the controller produces plans that reduce running times of 25 ECML HST trains by an average of 6.46 seconds by increasing the running times of 6 ECML commuter trains by an average of 10.79 seconds. Similarly, for Finsbury control area, 6 ECML commuter trains achieve average improvements of 12.04 seconds while 11 Hertford-Loop commuter trains see an increase of 3.13 seconds in terms of their mean running times inside the control area. Table 5.9 presents comparison of mean delays for all services between FCFS and ADP at major stations within our study network. London King's Cross station is the terminus of all ECML trains, Moorgate is the terminus for all Hertford-Loop trains, and Finsbury Park station is where all ECML commuter trains and most ECML HST services stop (15 services in the present case). The mean delays have been categorised by taking the mean value of delays of over 0, 1, 3 and 5 minutes. UK rail industry separates delays using a similar technique for different purposes. For instance, services arriving at their terminus station within 1 minute of their scheduled arrival times are monitored for the 'right time performance' measure; the government-regulated public performance measure (PPM) calculates the percentage of trains which arrive at their terminating station within 5 minutes of their scheduled arrival time [117]. Mean delays in Table 5.9 are calculated for the subset of trains that experience delays $> X$ minutes under either FCFS or ADP, or both.

The trend observed in mean running times within control areas can also be observed in

Table 5.9, where services that benefited from the re-scheduling (or the choice of objective function) have carried those improvements to their downstream planned station stops. As the threshold for considered delays increases, so does the difference in mean delays between FCFS and ADP. In the case of London King's Cross station, the difference between mean delays for delays over 0, 1, 3 and 5 minutes show improvements in favour of ADP of 9.6%, 10.4%, 15.3% and 26.3% respectively when compared to FCFS. The effect is reversed in the case of Moorgate where ADP results in delay increases of 15.0%, 11.91%, 4.4% and 9.4% for delays over 0, 1, 3 and 5 minutes respectively when compared to FCFS.

From the mean values, it may appear that delays have been pushed elsewhere in the network. However, given the number of trains terminating at London King's Cross (31 trains) and Moorgate (11 trains), it is clear that overall improvements have been achieved by applying ADP on this network.

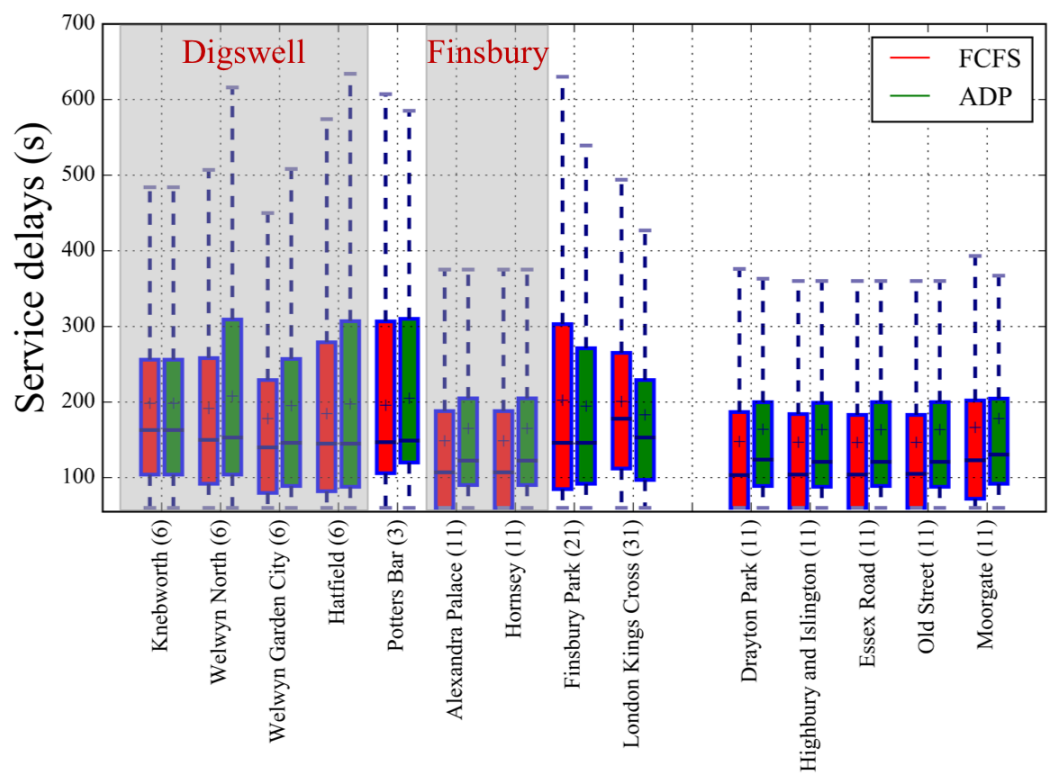


Figure 5.15: Comparison of individual delays of over 1 minute between FCFS and ADP for all trains stopping at stations inside or downstream of the control areas; numbers in brackets indicate total number of scheduled stops per simulation at the station, and + signs indicate mean values.

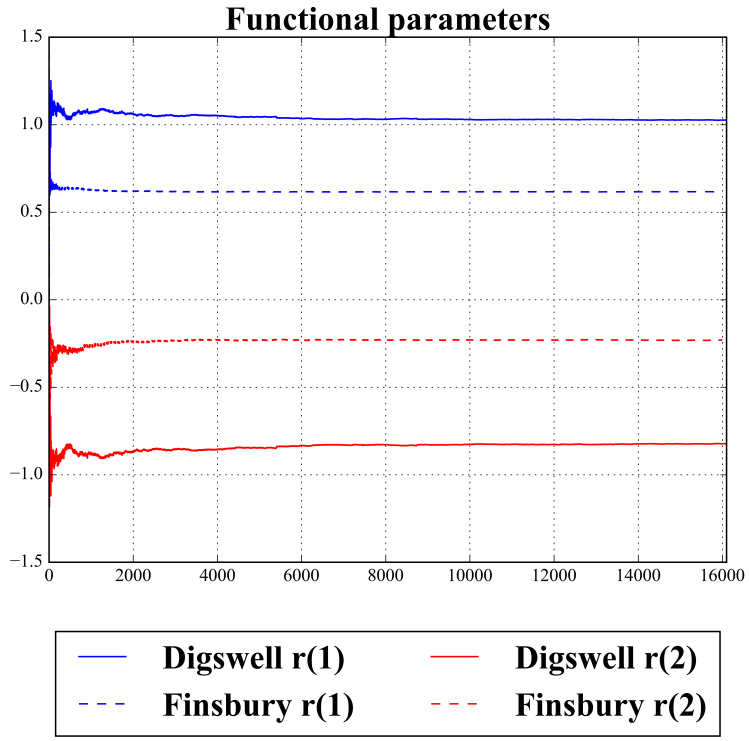


Figure 5.16: Evolution of LSTD functional parameters r for different control areas.

Figure 5.15 presents all arrival delays of over 1 minute for all stations within or downstream of the control areas. This shows effects at other stations within the network. The overall trend indicates more efficient operations in terms of time spent traversing the tracks in the control area (short term) which in turn translates into a reduction in delays at terminal stations (long term). This is evident in the service delays of Figure 5.15 which shows an increase in station delays in the vicinity of the control areas where slower trains from the control areas dwell, though this is reversed when all service delays are measured at terminal stations.

Figure 5.16 presents the evolution of functional parameters r for LSTD for Digswell and Finsbury control areas. Recalling that LSTD learning solves for these parameters by minimising the sum of squares of all temporal difference errors since the start of operation, similar to the previous Section, the r values stabilise early on and after around 1500 learning opportunities. The values for the functional parameters stabalise on different values for our control areas. This suggests that convergence of values is unique for the choice of

control area, timetable and other operational factors.

5.6 Summary and discussion

Having conducted numerical experiments using infrastructure and operational data for the East Coast Main Line (ECML), it is evident that ADP is capable of tackling the railway traffic management problem effectively and in a timely manner.

In Section 5.2, it was shown that the ADP approach is sensitive to the choice of objective function and unsuitable choice can result in poor performance. It was shown that, in the case of railway re-scheduling, objective functions may be outperformed by others in terms of their primary objective. Minimising consecutive delay was found to be the most effective in minimising consecutive delays, and minimising total sectional running times was found to be the most effective in minimising total delays of all trains in the control area.

Results of the numerical tests presented in Section 5.3 show that features must be selected carefully, as some combinations of them do not improve performance compared to a 'greedy' approach in which the criterion for optimisation focuses exclusively on short-term performance.

In Section 5.4, two traffic scenarios were used to simulate the stochastic railway environment, each equating to 2,000 morning peaks, to represent moderate and severe perturbations to trains running on the test network. As the primary measure, the objective function of mean consecutive delays was reduced by around 28% using the ADP framework. We also found that the parameters in TD and LSTD approach similar values with different mean train entry delays with the selected features. To approximate the value function, TD achieves smaller errors compared to LSTD in both traffic scenarios, and errors were smaller for both TD and LSTD in the traffic scenario with moderate perturbations compared to severe perturbations. We also found that adopting all adjustments to parameters r in one-step TD achieved better approximations, although this comes at the cost of reduced stability in the parameter estimates. The detailed evaluation of the cost during a single stage represented by $g(.)$ in equation (4.4) was found to be the most computationally expensive

part of the present ADP framework. While testing ADP in deterministic environment, computational times of 0.11 second per stage on average were achieved when $T = 3$, compared with 1.1 seconds per stage on average in stochastic environment when $T = 3$ due to the 30 simulations with draws from the distributions for each evaluation of expected performance according to equation (3.7).

To have a fully stochastic framework, this kind of railway traffic control could be formulated as a Markov decision process, given that all transition probabilities are known. Such a treatment would be comparable with application of the learning approaches described in this thesis. Therefore, the present ADP framework can be readily extended to include this capability.

In Section 5.5, the framework was deployed in a realistic simulation environment in a distributed setup across a large and dynamic network and controlled known critical areas of the ECML network and evaluated network-wide performance in deterministic simulations. Given that the penalty system for delayed trains in the UK is mainly concerned with delays at large or terminal station, the focus of this Section was analysis on the major stations within the study network. We found improvements in service delays at the largest station within the study network (London King's Cross) which was the terminal station for most of the timetable's services, where delays where improved by around 10-26% according to different rail industry performance measurements. This reduction in delays at the largest station directly related to increases of around 10-15% in delays at a less congested terminal station (Moorgate). Nevertheless, the overall performance was improved by around 6-15% according to different performance measurements, when delays of all trains at their terminal stations were considered . It is also worth noting that over the 3 hours and 45 minutes period of our delay scenarios, the controller has regained more than a minute of track occupancy time on average within Digswell control area. Given that the mean timetable running time of an ECML HST in this control area is around 3 and a half minutes, it may be worth investigating whether implementing a re-scheduler like the one presented in this thesis can facilitate addition of one more HST service in the ECML timetable per day.

As for the ADP performance in Section 5.5, the functional parameters r were observed to become stable after one thousand learning opportunities; in practice they could be calculated according to a period of training using historical data then implemented. Minimising total running times of services has resulted in improved performance on our study network, and further research is required to test and compare other more complex objective functions for ADP and on other networks of similar or higher complexity.

As an adaptive ADP approach for Railway Traffic Management (RTM), this study has focused on isolated junctions. The objective of RTM is to achieve good network-wide performance, so the ability of ADP to co-ordinate network traffic is an interesting topic for further research. The challenge here would be to represent traffic state of adjacent control areas effectively and hence extract appropriate features to use in the approximate performance function of the local ADP.

Chapter 6

Conclusions

"Oh threats of Hell and hopes of Paradise! One thing at least is certain - this life flies; one thing is certain and the rest is lies." - Omar Khayyam

6.1 Research summary

Given the increase in the volume of traffic on railway networks, and the advent of the "digital age", railway undertakings around the world are investing in intelligent railway systems that would manage operations in real-time, and so improve utilisation of the existing infrastructure to deliver a greater level of service to costumers. The research undertaken for this thesis contributes to this endeavour.

This thesis presents an investigation into an adaptive railway traffic controller for real-time operations that applies approximate dynamic programming (ADP) and reinforcement learning (RL). By assessing requirements and opportunities, the controller aims to limit train delays by advantageously controlling the sequencing of trains at critical locations in a timely manner.

Few previous authors have investigated methods that are appropriate for use in stochastic environments and use data generated by trains to learn the underlying process of railway traffic in real-time. Most approaches suffer from high computational complexity, which makes them inefficient or difficult to adopt for practical operation. This thesis has therefore developed an ADP framework to reduce the computational burden which in turn confers

flexibility in control in a stochastic environment that has been tested using a Monte-Carlo scheme.

The problem is formulated as a dynamic program and ADP is used to approximate the optimised value function, and RL techniques to update the approximation. This algorithmic framework reduces the number of states to be evaluated substantially, which leads to a corresponding reduction in the computational burden.

In this investigation, the variants of temporal difference (TD) and least-squares temporal difference (LSTD) learning techniques have been explored. Because number of features employed is sufficiently small, the parameters of the linear approximation function can be updated using a closed-form expression based on Newton's method. This avoids the widely used methods of TD learning based on gradient descent that have been found to lead to policy oscillation and overshooting.

The strongest advantages of Newton's method over gradient descent methods, relevant to the application of ADP, are:

- rapid convergence, and
- performance is not dependent on the choice of control parameters.

On the other hand, the main disadvantage is the computational cost of forming and inverting the Hessian matrix to solve each Newton step. However, this is only an issue for problems that require the use of large feature vectors and even then a possible alternative can be offered by a family of algorithms called *quasi-Newton methods*. These methods require less computational effort to form the search direction, but they share some of the principal advantages of Newton methods, such as the rapid convergence.

The present research has shown that features of the approximation function must be selected carefully, as some possible combinations do not improve performance compared to a 'greedy' approach in which the criterion for optimisation focuses exclusively on short-term performance. Therefore, appropriate feature engineering needs to be conducted to successfully configure ADP with integrated RL.

The evaluations to test ADP performance in stochastic environment reported in this thesis were based on two traffic scenarios, each equating to 2,000 morning peaks, and represent moderate and severe perturbations to trains running on the test network. As the primary measure, the objective function of mean consecutive delays was reduced by around 28% using the ADP framework by comparison with conventional first-come first-served (FCFS) control. The even greater proportionate reduction in the higher percentile points of the distribution of consecutive delays shows that the ADP framework reduces longer delays effectively, helping to improve regulation of train services.

Testing the ADP framework in a realistic simulation environment in a distributed setup across a large and dynamic network by controlling known critical areas of the network was also evaluated with particular emphasis on the network-wide performance. Given that the current penalty system for delayed trains in the UK is mainly concerned with delays at large or terminal stations, the focus of analysis was on the major stations within the study network. Improvements were found in service delays at the largest station within the study network (London King's Cross) which was the terminal station for most of the timetable services, where delays were improved by around 10-26% according to different rail industry performance measurements. This reduction in delays at the largest station directly related to increases of around 10-15% in delays at a less congested terminal station (Moorgate). Nevertheless, the overall performance was improved by around 6-15% according to different performance measurements, when delays of all trains at their terminal stations were considered.

Comparison between the Newton-based TD and LSTD approaches for estimation of parameters in the value approximation function, similar values between these approaches were found, though the values estimated by one-step TD fluctuated from moment to moment whilst those estimated by LSTD necessarily stabilised because they are calculated using all earlier results with equal weighting. The TD estimates can be stabilised to some degree by adopting a weighting with exponential decay, so focusing on the most recent calculations. The error in approximating the value function using TD is smaller compared to

LSTD in both traffic scenarios, and errors were smaller for both TD and LSTD in the traffic scenario with moderate perturbations compared to severe perturbations. It was also found that adopting all adjustments to parameters of the approximation function in one-step TD achieved better approximations, although at the cost of reduced stability in the parameter estimates.

The detailed calculation of the cost during a single stage was found to be the most computationally expensive part of the present ADP framework. While testing ADP in a deterministic environment a computational times of 0.07 second per stage on average for a lookahead horizon of $T = 3$ trains was achieved, compared with 1.1 seconds per stage on average in stochastic environment due to the 30 simulations with draws from the distributions for each evaluation of expected performance. This contrasts with the computational time required for FCFS, which was 0.016 second per stage in the experiments.

6.2 Future work

Over the 3 hours and 45 minutes period of the delay scenarios in Chapter 5, the controller regained more than a minute of track occupancy time on average within the control area. Given that the mean timetable running time of an ECML HST in the control area is around 3 and a half minutes, it may be worth investigating whether implementing a re-scheduler same as the one presented in this thesis can facilitate addition of one more HST service in the ECML timetable per day.

The adaptive ADP approach of this thesis has focused on traffic management of isolated conflict points (*i.e.* junctions, crossings and merges). The objective of railway traffic management (RTM) is to achieve improved network-wide performance, so the ability of ADP to co-ordinate traffic of a number of control areas is an interesting topic for further research. The challenge would be to represent traffic state of adjacent control areas effectively and hence extract appropriate features to use in the approximate performance function of the local ADP. It may be necessary to use a nonlinear function approximation, such as Artificial Neural Networks similar to the ADP framework in [118], to configure ADP for network-

wide co-ordination of traffic.

Configuring ADP with linear function approximation to control and co-ordinate a large number of control areas may require use of large feature vectors. A concern associated with this would be the computational effort required to invert the Hessian matrix of the Newton-based TD learnings discussed in this thesis. Therefore, the application of quasi-Newton methods, which are computationally efficient, could be investigated in future works.

In this investigation, it is shown that ADP improves operational performance when applied in a distributed configuration and it would be interesting to investigate how co-ordination improves on performance relative to the findings of this thesis.

The stochastic approach in this thesis employs a Monte-Carlo scheme to deal with the uncertainty of railway operations. To have a fully stochastic framework, the railway traffic control approach of this thesis could be formulated as a Markov decision process, given that all transition probabilities are known. Such a treatment would be comparable with application of the learning approaches described in this thesis. Therefore, the present ADP framework can be readily extended to include this capability.

Even though Monte-Carlo simulation is an effective method to make decisions in noisy environments, as was discussed in the previous Section, it significantly adds to the computational time required to compute control decisions. A good direction of research could include exploration of methods to improve on this aspect of the ADP approach presented in this thesis.

The main distinguishing factor of the present ADP framework is its ability to exploit real-time data to facilitate optimised asset operations on railway networks. The framework presented in this thesis uses data generated by trains to learn the processes underlying railway traffic in real-time to optimise signalling decisions. It does not guarantee optimality of solutions and trades off practicality over optimality. Little of the literature available on RTM considers data-based approaches to solve this complex optimisation. Further research in data-based techniques to build on the findings of this thesis would establish their applicability to traffic management of large railway networks.

The ADP framework presented in this thesis is a systematic and self-tuning approach which achieves superior performance compared to FCFS by considering the consequence of decisions in a timely manner and therefore is better suited for practical implementation. It automatically adapts parameters r to the objective function and the prevailing traffic on the rail network, allowing for better usage of real-time data generated by trains. Our investigation shows the potential of such approaches, which need to be further explored to realise a future where intelligent transport systems aid integrated and seamless transport.

Given the recent successes of reinforcement learning in the field of computer science [119, 120], demonstrating the capability of RL to control complex problems, and the findings of this thesis, investigation of a model-free RL approach to control railway traffic is warranted. A model-free RL approach would eliminate the need for a Monte-Carlo scheme in RTM and potentially result in a even more efficient approach as the one developed in this thesis.

Even though the ADP formulations presented can theoretically also consider moving block operations under, for instance, ETCS level 3, the evaluations of this thesis only include fixed block operations. Assessment of the present formulations under moving block signalling could be considered in future works.

Finally, the ADP approach of the present thesis has focused on optimising capacity utilisation by means of controlling order of trains at conflict points. Addition of other considerations such as passenger flows at stations and trajectory optimisation of the trains under consideration would potentially improve the findings of this investigation even further.

References

[1] Department for Transport. Rail Technical Strategy 2007. Technical report, DfT, London, United Kingdom, 2007.

[2] Department for Transport. Delivering a Sustainable Railway. Technical report, DfT, London, United Kingdom, 2007.

[3] Daniel Woodland. *Optimisation of Automatic Train Protection Systems*. PhD thesis, University of Sheffield, 2004.

[4] Andrea D'Ariano. *Improving Real-Time Train Dispatching: Models, Algorithms and Applications*. PhD thesis, Department of Transport & Planning, Faculty of Civil Engineering and Geosciences. Delft University of Technology, 2008.

[5] T. K. Ho, J. P. Norton, and C. J. Goodman. Optimal traffic control at railway junctions. In *Electric Power Applications*, volume 144, pages 140–148. IEE, 1997.

[6] Johanna Törnquist and Jan A Persson. N-tracked railway traffic re-scheduling during disturbances Johanna To. *Transportation Research Part B: Methodological*, 41(3):342–362, 2007.

[7] T. K. Ho and T. H. Yeung. Railway junction conflict resolution by genetic algorithm. *Electronics Letters*, 36(8):771–772, April 2000.

[8] Hugo P. Simão, Jeff Day, Abraham P. George, Ted Gifford, John Nienow, and Warren B. Powell. An Approximate Dynamic Programming Algorithm for Large-Scale

Fleet Management: A Case Application. *Transportation Science*, 43(2):178–197, 2009.

[9] Chen Cai, Chi Kwong Wong, and Benjamin G. Heydecker. Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 17(5):456–474, 2009.

[10] J. C. Medina, A. Hajbabaie, and R. F. Benekohal. A comparison of approximate dynamic programming and simple genetic algorithm for traffic control in oversaturated conditions — case study of a simple symmetric network. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1815–1820, Oct 2011.

[11] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[12] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2011.

[13] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 2nd edition, 1998.

[14] Alborz Geramifard. *Practical Reinforcement Learning Using Representation Learning and Safe Exploration for Large Scale Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 2012.

[15] John N Tsitsiklis and Benjamin Van Roy. An Analysis of Temporal-Difference Learning with Function Approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

[16] Valentina Cacchiani, Dennis Huisman, Martin Kidd, Leo Kroon, Paolo Toth, Lucas Veelenturf, and Joris Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, 2014.

[17] Francesco Corman and Lingyun Meng. A Review of Online Dynamic Models and Algorithms for Railway Traffic Management. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1274–1284, 2015.

[18] W. Fang, S. Yang, and X. Yao. A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2997–3016, Dec 2015.

[19] Carlo Mannino and Alessandro Mascis. Optimal Real-Time Traffic Control in Metro Stations. *Operations Research*, 57(4):1026 – 1039, 2009.

[20] F Mehta, C Rößiger, and M Montigel. Latent energy savings due to the innovative use of advisory speeds to avoid occupation conflicts. In *Computers in Railways XII*, volume 114, pages 99–108. WIT Transactions on The Built Environment, 2010.

[21] Ralf Borndörfer, Torsten Klug, Leonardo Lamorgese, Carlo Mannino, Markus Reuther, and Thomas Schlechte. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C: Emerging Technologies*, 74:196–211, 2017.

[22] Joseph P Greenway and Ralph H Sheldon. Train Control and Communications for Washington Metro. *Communications Society*, 12(6), 1974.

[23] C. S. Chang. Automatic train regulation system-a new concept in computerization of train control system. In *IEEE TENCON'90: 1990 IEEE Region 10 Conference on Computer and Communication Systems. Conference Proceedings*, pages 821–826 vol.2, Sept 1990.

[24] Ning Zhao. *Railway Traffic Flow Optimisation with Differing Control Systems*. PhD thesis, University of Birmingham, 2013.

[25] Ziyi Jiang. *Digital Route Model Aided Integrated Satellite Navigation and Low-cost Inertial Sensors for High-Performance Positioning on the Railways*. PhD thesis, UCL, 2010.

[26] David Kerr and Rowbotham Tony. *Introduction to railway signalling*. Institution of Railways Signal Engineers, London, 2001.

[27] Jörn Pachl. Application of blocking time analysis for specific signal arrangements. Transportation Research Board. 84th Annual Meeting on January 9-13, 2005.

[28] G. Xie, A. Asano, S. Takahashi, and H. Nakamura. Study on formal specification of automatic train protection and block system for local line. In *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement - Companion*, pages 35–40, June 2011.

[29] Rail Safety and Standards Board. GE/GN8605 - ETCS System Description, 2010.

[30] UNISIG Consortium. FFFIS for Eurobalise. Technical Report SUBSET-036, 2007.

[31] Peter Stanley. *ETCS for engineers*. Eurailpress; © IRSE, Institution of Railway Signal Engineers, Hamburg, 2011.

[32] Aris Pavlides. *Cost Functions for Railway Operations and Their Application to Timetable Optimisation*. PhD thesis, UCL, 2016.

[33] Lei Chen. *Real Time Traffic Management in Junction Areas and Bottleneck Sections on Mainline Railways*. PhD thesis, University of Birmingham, 2012.

[34] Linsha Dai. *Intelligent Real-time Train Rescheduling Management for Railway System*. PhD thesis, University of Birmingham, 2016.

[35] M. Hyoudou, T. Seto, Y. Matsuura, K. Komaya, T. Fukawa, and K. Morihara. An expert system for train operation adjustment of shinkansen. *IFAC Proceedings Volumes*, 30(8):1145 – 1150, 1997. 8th IFAC/IFIP/IFORS Symposium on Transportation Systems 1997 (TS '97), Chania, Greece, 16-18 June.

[36] Alexander Fay. A fuzzy knowledge-based system for railway traffic control. *Engineering Applications of Artificial Intelligence*, 13(6):719–729, 2000.

[37] Yung-Hsiang Cheng and Li-An Yang. A fuzzy petri nets approach for railway traffic control in case of abnormality: Evidence from taiwan railway system. *Expert Systems with Applications*, 36(4):8040 – 8048, 2009.

[38] M. Kuhn. Automatic route setting integrated in a modern traffic management system. In *1998 International Conference on Developments in Mass Transit Systems Conf. Publ. No. 453)*, pages 140–145, April 1998.

[39] A. D. Middelkoop and L. Loeve. Simulation of traffic management with FRISO. In *Computers in Railways X*, volume 88, pages 501–509. WIT Transactions on The Built Environment, 2006.

[40] Quan Lu, Maged Dessouky, and Robert C. Leachman. Modeling train movements through complex rail networks. *ACM Trans. Model. Comput. Simul.*, 14(1):48–75, January 2004.

[41] P. Weston, C. Goodman, R. Takagi, C. Bouch, and J. Armstrong. Minimising train delays in a mixed traffic railway network with consideration of passenger delay. In *7th World Congress on Railway Research*, June 2006.

[42] Tomii Norio, Tashiro Yoshiaki, Tanabe Noriyuki, Hirai Chikara, and Muraki Kunimitsu. Train rescheduling algorithm which minimizes passengers' dissatisfaction. In Moonis Ali and Floriana Esposito, editors, *Innovations in Applied Artificial Intelligence*, pages 829–838, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[43] Satoshi Kanai, Koichi Shiina, Shingo Harada, and Norio Tomii. An optimal delay management algorithm from passengers' viewpoints considering the whole railway network. *Journal of Rail Transport Planning & Management*, 1(1):25–37, 2011. Robust Modelling of Capacity, Delays and Rescheduling in Regional Networks.

[44] Marco Luethi, Giorgio Medeossi, and Andrew Nash. Structure and simulation evaluation of an integrated real-time rescheduling system for railway networks. *Networks and Spatial Economics*, 9(1):103–121, Mar 2009.

[45] Amie Albrecht, Phil Howlett, Peter Pudney, Xuan Vu, and Peng Zhou. Optimal driving strategies for two successive trains on level track subject to a safe separation condition. In *2015 American Control Conference (ACC)*, pages 2924–2929, Chicago, IL, 2015. IEEE.

[46] Fang Xu, Benjamin Heydecker, Andy H F Chow, and Taku Fujiyama. Optimisation Framework for Rail Traffic Regulation at a Single Junction. In *Proceedings of the 6th International conference on railway operations modelling and analysis*, Chiba Institute of Technology, Japan, 2015, 2015.

[47] M Luethi. Evaluation of energy saving strategies in heavily used rail networks by implementing an integrated real-time rescheduling system. *Computers in Railways XI*, 103:349–358, 2008.

[48] Francesco Corman. *Real-time Railway Traffic Management: dispatching in complex, large and busy railway networks*. PhD thesis, Delft University of Technology, 2010.

[49] Andrea D'Ariano and Marco Pranzo. An Advanced Real-Time Train Dispatching System for Minimizing the Propagation of Delays in a Dispatching Area Under Severe Disturbances. *Networks and Spatial Economics*, 9:63–84, 2009.

[50] Johanna Törnquist. Railway traffic disturbance management - An experimental analysis of disturbance complexity , management objectives and limitations in planning horizon. *Transportation Research Part A: Policy and Practice*, 41(3):249–266, 2007.

[51] Lingyun Meng and Xuesong Zhou. Robust single-track train dispatching model under a dynamic and stochastic environment : A scenario-based rolling horizon solution approach. *Transportation Research Part B: Methodological*, 45(7):1080–1102, 2011.

[52] Egidio Quaglietta, Francesco Corman, and Rob M.P. Goverde. Stability analysis of railway dispatching plans in a stochastic and dynamic environment. *Journal of Rail Transport Planning and Management*, 3(4):137–149, 2013.

[53] B. Adenso-Diaz, M. Oliva-Gonzalez, and P. Gonzalez-Torre. On-line timetable rescheduling in regional train services. *Transportation Research Part B: Methodological*, 33(6):387–398, 1999.

[54] Johanna Törnquist. Design of an effective algorithm for fast response to the rescheduling of railway traffic during disturbances. *Transportation Research Part C: Emerging Technologies*, 20(1):62–78, 2012.

[55] Sai Prashanth Josyula, Johanna Törnquist Krasemann, and Lars Lundberg. A parallel algorithm for train rescheduling. *Transportation Research Part C: Emerging Technologies*, 95:545–569, 2018.

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

[57] Paola Pellegrini, Grégory Marlière, and Joaquin Rodriguez. Real time railway traffic management modeling track-circuits. In *ATMOS12*, pages 23–34, 2012.

[58] Yun-Hong Min, Myoung-Ju Park, Sung-Pil Hong, and Soon-Heum Hong. An appraisal of a column-generation-based algorithm for centralized train-conflict resolution on a metropolitan railway network. *Transportation Research Part B: Methodological*, 45(2):409–429, 2011.

[59] R. Acuna-Agost. *Mathematical modeling and methods for rescheduling trains under disrupted operations*. PhD thesis, Université d'Avignon, 2010.

[60] Y. Cui. *Simulation-based hybrid model for a partially automatic dispatching of railway operation*. PhD thesis, University of Stuttgart, 2010.

[61] Francesco Corman, Dario Pacciarelli, Andrea D'Ariano, and Marcella Samà. Rescheduling railway traffic taking into account minimization of passengers' discomfort. In Francesco Corman, Stefan Voß, and Rudy R. Negenborn, editors, *Computational Logistics*, pages 602–616. Springer International Publishing, 2015.

[62] Alessandro Mascis and Dario Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.

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

[64] Andrea D'Ariano, Marco Pranzo, and Ingo A I.A. Hansen. Conflict Resolution and Train Speed Coordination for Solving Real-Time Timetable Perturbations. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):208–222, 2007.

[65] Maura Mazzarello and Ennio Ottaviani. A traffic management system for real-time traffic optimisation in railways. *Transportation Research Part B: Methodological*, 41(2):246–274, 2007.

[66] Pavle Kecman, Francesco Corman, Andrea D Ariano, and Rob M P Goverde. Rescheduling models for railway traffic management in large-scale networks. *Public Transport*, 5(1):95–123, 2013.

[67] Francesco Corman, Andrea D Ariano, Dario Pacciarelli, and Marco Pranzo. Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C: Emerging Technologies*, 20(1):79–94, 2012.

[68] Francesco Corman, Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44(1):175–192, 2010.

[69] F Corman, A D Ariano, D Pacciarelli, and M Pranzo. Dispatching and coordination in multi-area railway traffic management. *Computers and Operation Research*, 44:146–160, 2014.

[70] Francesco Corman, Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport*, 2(3):219–247, 2010.

[71] F Corman, A D'Ariano, M Pranzo, and I A Hansen. Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transportation Planning and Technology*, 34:341–362, 2011.

[72] Christian Strotmann. *Railway scheduling problems and their decomposition*. PhD thesis, Osnabruck University, 2007.

[73] F Corman and E Quaglietta. Closing the loop in real-time railway control : Framework design and impacts on operations. *Transportation Research Part C: Emerging Technologies*, 54:15–39, 2015.

[74] D. E. Brown, C. L. Huntley, B. P. Markowicz, and D. E. Sappington. Rail network routing and scheduling using simulated annealing. In *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 589–592 vol.1, Oct 1992.

[75] T. K. Ho and T. H. Yeung. Railway junction traffic control by heuristic methods. *IEE Proceedings - Electric Power Applications*, 148(1):77–84, Jan 2001.

[76] D. Šemrov, R. Marseti˘c, M. Žura, L. Todorovski, and A. Srdic. Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological*, 86:250–267, 2016.

[77] Jiateng Yin, Tao Tang, Lixing Yang, Ziyou Gao, and Bin Ran. Energy-efficient metro train rescheduling with uncertain time-variant passenger demands : An approximate

dynamic programming approach. *Transportation Research Part B: Methodological*, 91:178–210, 2016.

[78] L Chen, F Schmid, M Dasigi, B Ning, C Roberts, and T Tang. Real-time train rescheduling in junction areas. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 224(6):547–557, 2010.

[79] C. Hirai, N. Tomii, Y. Tashiro, S. Kondou, and A. Fujimori. An algorithm for train rescheduling using rescheduling pattern description language R. In *Computers in Railways X*, volume 88, pages 551–561. WIT Transactions on The Built Environment, 2006.

[80] Bart Kersbergen, Ton van den Boom, and Bart De Schutter. Distributed model predictive control for railway traffic management. *Transportation Research Part C: Emerging Technologies*, 68:462–489, 2016.

[81] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer-Verlag New York, New York, 2012.

[82] Maite Pena Alcaraz, Mort Webster, and Andres Ramos. An approximate dynamic programming approach for designing train timetables. Technical report, Engineering Systems Division, Massachusetts Institute of Technology, 2011.

[83] Warren B Powell and Belgacem Bouzaiene-ayari. Approximate dynamic programming for rail operations. In Christian Liebchen Mesa, Ravindra K. Ahuja, and Juan A., editors, *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07)*, pages 191–208, Dagstuhl, Germany, 2007.

[84] Chen Cai. *Adaptive traffic signal control using approximate dynamic programming*. PhD thesis, UCL, 2009.

[85] Biao Yin, Mahjoub Dridi, and Abdellah El Moudni. Approximate Dynamic Programming with Recursive Least-Squares Temporal Difference Learning for Adap-

tive Traffic Signal Control. In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, pages 3463–3468, Osaka, Japan, 2015.

[86] Michael T Davis, Matthew J Robbins, and Brian J Lunday. Approximate dynamic programming for missile defense interceptor fire control. *European Journal of Operational Research*, 259(3):873–886, 2017.

[87] Warren B Powell, Hugo P Simao, and Belgacem Bouzaiene-ayari. Approximate Dynamic Programming in Transportation and Logistics : A Unified Framework. *European Journal of Transportation and Logistics*, 1(3):237–284, 2012.

[88] Katerina P Papadaki and Warren B Powell. Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem. *European Journal of Operational Research*, 142(1):108–127, 2002.

[89] Verena Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research*, 219(3):611–621, 2012.

[90] Haitao Li and Norman K. Womer. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, 246(1):20–33, 2015.

[91] W Zhang and T Dietterich. A Reinforcement Learning Approach to Job-Shop Scheduling. In *IJCAI'95*, pages 1114–1120, Montreal, Canada, 1995.

[92] José A Ramírez-Hernández and Emmanuel Fernandez. An Approximate Dynamic Programming Approach for Job Releasing and Sequencing in a Reentrant Manufacturing Line. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 201–208, Honolulu, Hawaii, USA, 2007.

[93] Aditya Medury and Samer Madanat. Incorporating network considerations into pavement management systems : A case for approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 33:134–150, 2013.

[94] B Van Roy, D P Bertsekas, Y Lee, and J N Tsitsiklis. Neuro-Dynamic Programming Approach to Retailer Inventory Management. In *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, CA, USA, USA, 1997. IEEE.

[95] Tatpong Katanyukul, William S. Duff, and Edwin K. P. Chong. Approximate dynamic programming for an inventory problem: empirical comparison. *Computers & Industrial Engineering*, 60(4):719–743, 2011.

[96] Katerina P. Papadaki and Warren B. Powell. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. Technical Report 7, 2003.

[97] Hans-Jörg Schütz and Rainer Kolisch. Approximate dynamic programming for capacity allocation in the service industry. *European Journal of Operational Research*, 218(1):239–250, 2012.

[98] Dimitri J. Papageorgiou, Myun-Seok Cheon, George Nemhauser, and Joel Sokol. Approximate Dynamic Programming for a Class of Long-Horizon Maritime Inventory Routing Problems. *Transportation Science*, 49(4):870–885, 2014.

[99] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic programming*. Athenas Scientific, Belmont, MA, 1995.

[100] Richard S Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3(1):9–44, 1988.

[101] Christopher J Watkins. Q-Learning. *Journal of Machine Learning Research*, 8(3):279–292, 1992.

[102] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, 1994.

[103] Shalabh Bhatnagar, Mohammad Ghavamzadeh, Mark Lee, and Richard S Sutton. Incremental natural actor-critic algorithms. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 105–112. Curran Associates, Inc., 2008.

[104] David Silver. *Reinforcement Learning and Simulation-Based Search in Computer Go*. PhD thesis, University of Alberta, 2009.

[105] Benjamin Van Roy. *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, Cambridge, MA, USA, 1998.

[106] Dimitri P Bertsekas. Approximate policy iteration : a survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

[107] Paul Wagner. Policy oscillation is overshooting. *Neural Networks*, 52:43–61, 2014.

[108] Steven J Bradtke and Andrew G Barto. Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996.

[109] Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, Nov 2002.

[110] M. Pang, W. Jiang, and J. Li. Analysis about impact of gsm-r cell handoff on data transfer delay in china train control system. In *2010 International Conference on Electronics and Information Engineering*, volume 2, pages 149–153, Aug 2010.

[111] E. R. Petersen and A. J. Taylor. A structured model for rail line simulation and optimization. *Transportation Science*, 16:625–643, 1982.

[112] Maged M. Dessouky and Robert C. Leachman. A simulation modeling methodology for analyzing large complex rail networks. *Simulation*, 65(2):131–142, 1995.

[113] David Kirkwood. *Simulation, test and performance evaluation of railway control strategies and algorithms*. PhD thesis, University of Birmingham, 2014.

[114] A. Radtke. Handling of railway operation problems with railsys. In *WCRR-Proceedings, Köln*, 2001.

[115] S. Lu, S. Hillmansen, and C. Roberts. A power-management strategy for multiple-unit railroad vehicles. *IEEE Transactions on Vehicular Technology*, 60(2):406–420, 2011.

[116] B P Rochard and F Schmid. A review of methods to measure and calculate train resistances. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 214(4):185–199, 2000.

[117] Office of Rail and Road. Passenger & Freight Rail Performance: Quality and Methodology Report. Technical report, London, United Kingdom, 2018.

[118] Fang He, Jie Yang, and Meng Li. Vehicle scheduling under stochastic trip times: An approximate dynamic programming approach. *Transportation Research Part C: Emerging Technologies*, 96:144 – 159, 2018.

[119] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, and Laurent Sifre. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[120] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Appendix A

Multivariate Newton's method

For $x \in \mathbf{dom}\ f$, a *Newton step* Δx_{nt} (for f, at x) is defined as the vector

$$\Delta x_{nt} = -\nabla^2 f(x)^{-1} \nabla f(x), \tag{A.1}$$

and the Hessian matrix $\mathbf{H}$, of a twice continuously differentiable function f at a point $x = (x_1, \cdots, x_n)$, is the matrix of second partial derivatives

$$\mathbf{H}(x) = \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}. \tag{A.2}$$

To derive the multivariate Newton's method, take the second-order Taylor expansion around a point x_k

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \mathbf{H}(x_k)(x - x_k). \tag{A.3}$$

This is a convex quadratic function of $(x - x_k)$, and is minimised when $(x - x_k) = \Delta x_{nt}$. Thus, the Newton step Δx_{nt} is what should be added to the point x to minimise the second-order approximation of f at x.

Supposing that $\mathbf{H}(x_k)$ is positive definite, the unique point that minimises the quadratic form

on the right hand side is therefore

$$x = x_k - \mathbf{H}(x_k)^{-1}\nabla f(x_k). \tag{A.4}$$

Appendix B

List of symbols

s is the vector of state variables

S is the state space

u is the decision variable

U is the decision space

$g(.)$ is the one step cost function

t is the single step

α is the future-discount factor

θ is the discount rate

$f(.)$ is the state transition function

v is random information

E is the expectation over random information

J^* is the optimal performance

P is the matrix of transition probabilities

T is the time horizon

π is the policy or the sequence of decisions u

$\widetilde{J}$ is an observation of the optimal value function

$\widehat{J}$ is the approximation of the optimal value function

r is the vector of parameters of the approximation function

$\phi(.)$ is the features extraction function

r^* is the unique optimal parameter vector

Δr is the adjustments to r

Q^π is the decision-value function for policy π

$\rho(s,u)$ is the preference of taking decision u in state s

τ regularises decision making

ω is a vector of parameters of an approximation function

ψ is a features extraction function

Π is the projection operator

$\langle .,. \rangle$ is the inner product operator

δ is the TD-error

$\beta_i (0 \leq i \leq t)$ is the positive discounting factors that are non-increasing in i

e is an error term

η regularises adjustments to r

$\mathbf{H}$ is the Hessian matrix

γ regularises past information

φ is the vector of step changes in ϕ