Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 38 (2019) 201–221

Transportation Research Procedia

www.elsevier.com/locate/procedia

23rd International Symposium on Transportation and Traffic Theory, ISTTT 23, 24-26 July 2019, Lausanne, Switzerland

Adaptive Railway Traffic Control using Approximate Dynamic Programming

Taha Ghasempour[a,*], Benjamin Heydecker[a,*]

[a]*Centre for Transport Studies, University College London, London WC1E 6BT, United Kingdom*

Abstract

This study presents an adaptive railway traffic controller for real-time operations based on approximate dynamic programming (ADP). By assessing requirements and opportunities, the controller aims to limit consecutive delays resulting from trains that entered a control area behind schedule by sequencing them at critical locations in a timely manner, thus representing the practical requirements of railway operations. This 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 reinforcement learning techniques. By using this approximation, the ADP avoids extensive explicit evaluation of performance and so reduces the computational burden substantially. In this investigation, we explore formulations of the approximation function and variants of the learning techniques used to estimate it. Evaluation of the ADP methods in a stochastic simulation environment shows considerable improvements in consecutive delays by comparison with the current industry practice of First-Come-First-Served sequencing. We also found that estimates of parameters of the approximate value function are similar across a range of test scenarios with different mean train entry delays.

Peer-review under responsibility of the scientific committee of the 23rd International Symposium on Transportation and Traffic Theory.

Keywords: Approximate Dynamic Programming; Railway Traffic Management; Adaptive Control; Reinforcement Learning.

* Corresponding author. Tel.: +44-2076791553; fax: +44-2076793042.
E-mail address: taha.ghasempour.14@ucl.ac.uk (T. Ghasempour); b.heydecker@ucl.ac.uk (B. Heydecker)

2352-1465
Peer-review under responsibility of the scientific committee of the 23rd International Symposium on Transportation and Traffic Theory.
10.1016/j.trpro.2019.05.012

1. Introduction

Today's cities are becoming larger and ever more congested. Railways are now more popular than ever before as means of commuting into (or out of) cities and passenger numbers are predicted 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 decreasing 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 (Department for Transport, 2007a, 2007b).

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, 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. Literature review

A growing literature is available on real-time RTM (Cacchiani et al., 2014; Corman and Meng, 2015; Fang et al., 2015) which has shown the effectiveness and benefits of using such tools in some particular circumstances.

Here we distinguish between different types of delays according to the following definitions:

i) *primary delays* arise from deviations from normal operation traffic, e.g. longer than expected dwell times,

ii) *consecutive delays* are ones caused to other trains from the primary delays, and

iii) *total delays* are the sum of all primary and consecutive delays.

Adenso-Diaz et al. (1999) formulated the real-time timetable and rolling stock re-scheduling problem in cases of large disruption to one or more trains as a Mixed Integer Linear Programme (MILP). Because the problem is hard to solve using existing MILP solvers, a heuristic was developed to produce feasible solutions that are intended to increase the number of passengers transported within the planning period. The model was implemented in practice and solutions produced were reported to be appealing. Törnquist and Persson (2007) also formulated the re-scheduling problem for large networks as a MILP model which considers reordering and rerouting of trains with the objective of minimising train delays. Because MILP solvers could not find feasible solutions within reasonable times, Törnquist (2012, 2007) extended this work by presenting a heuristic and reported that good solutions could be calculated quickly.

Another instance of MILP formulation for re-scheduling was formulated by Pellegrini et al. (2014) with the aim of obtaining optimal solutions for re-scheduling and re-routing of trains at a local level. Two objectives were considered: i) minimising the largest consecutive delay, and ii) minimising total consecutive delays for all trains. A rolling horizon approach was adopted to evaluate the effects of using different planning periods.

The alternative graph formulation for railway re-scheduling has also been widely reported. This is a discrete optimisation formulation which can be used to model re-scheduling problems with no-wait and no-store constraints that has been applied to job shops (Mascis and Pacciarelli, 2002). Several works are reported in the literature using

the Alternative graph model to formulate the railway re-scheduling problem; among the first, D'Ariano et al. (2007a) proposed a branch-and-bound solution algorithm for re-scheduling trains in real-time, where block sections are characterised as machines and trains as jobs in the formulation, therefore providing a microscopic view of the railway network under consideration. This formulation allows for presentation of railway network safety constraints including that at any time, at most one train is present in each block section of the track. Building on their previous work D'Ariano et al. (2007b) adopted a blocking time model to evaluate the feasibility of headways between following trains, and train speed profiles are updated considering preceding signal aspects. Mazzarello and Ottaviani (2007) similarly use the Alternative graph formulation to produce a conflict-free timetable for trains; they also computed associated train speed profiles to minimise train energy consumption.

Corman et al. (2012, 2010a, 2009) extended the work of D'Ariano et al. (2007a) to include local re-routing of trains with the objective of minimising the maximum consecutive delay. Corman et al. (2014, 2010b) extended their work by introducing a controller that coordinates 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.

Ho et al. (1997) 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. Their work established DP as a method that is appropriate to model railway junction traffic but not ideal to derive optimal solutions in real-time due to the computational effort required.

Among the few that have investigated and tested RTM methods in stochastic environments, Meng and Zhou (2011) 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. (2013) also tested a re-scheduling approach, formulated using an alternative graph model, under uncertainty by employing a Monte-Carlo scheme. Larsen et al. (2013) proposed a framework to evaluate the robustness of a re-scheduling solution according to its accommodation of small stochastic variations in running and dwell times of trains in the network without increasing output delays; this research showed that allowing for the stochastic nature of the railway environment improves performance compared to the first-come-first-served (FCFS) heuristic.

1.2. Motivations and objectives

The available literature on RTM shows that comparatively few industrial prototypes of these systems have been implemented in practice (Borndörfer et al., 2017; Mannino and Mascis, 2009; Mehta et al., 2010). 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 complex 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 congested and complex railway networks calculating optimal solutions could be computationally intractable due to the number of cases to be considered. 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 accuracy 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.

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 (Li and Womer, 2015; Medury and Madanat, 2013; Papadaki and Powell, 2002; Papageorgiou et al., 2014; Simão et al., 2009; Van Roy et al., 1997; Zhang and Dietterich, 1995). ADP is a variant of DP which uses approximation in the evaluation of candidate control decisions. It is often used when the robustness of a DP is required but the problem is too complicated for a DP strategy to tackle in an efficient and timely manner. 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.

Although the use of ADP to solve operational railway problems is rare, studies in railway related ADP are emerging (Powell and Bouzaiene-ayari, 2007; Šemrov et al., 2016; Yin et al., 2016). An area of research which has similarities to the RTM problem is the field of road traffic signal control. Cai et al. (2009) investigated the application of ADP to signal control of road traffic, aiming to develop an adaptive controller for online operation. In this work, a linear approximation function was employed with parameter values updated during operation using reinforcement learning techniques. Two learning techniques were used in this work: Temporal-Difference (TD) learning (Sutton, 1988) and perturbation learning (Papadaki and Powell, 2003). The TD method tracks the difference between current approximation of performance and values estimated from current state values. This difference is used to update values of the parameters in the approximation function. Perturbation learning estimates the gradients of the approximation function by perturbing the system state. Despite these different learning methods, Cai et al. found no substantial difference between their performance in numerical experiments.

Given the robustness of ADP in tackling scheduling problems, as is presented in the literature, the objective of this paper is to develop an adaptive RTM framework, based on ADP with two distinct learning techniques, to mange railway traffic by controlling train sequences at critical points (such as junctions, merges and crossings). The framework is then evaluated in a separate high fidelity stochastic simulation environment to investigate its performance in the presence of uncertainties that are typical of practical operation.

The remainder of this paper is organised as follow: In section 2 we introduce ADP and present our learning methods in general terms. The ADP frameworks are then developed for the railway re-scheduling problem in section 3. Section 4 introduces our test case network, discusses the stochastic environment in which our ADP frameworks are tested and presents findings from our numerical experiments. We conclude this paper and consider the scope for future research in section 5.

Nomenclature

s is a vector of system state;	E represents the expectation over random information;
$J(s)$ is the true value function associated with state s;	α is a discount factor;
$\hat{J}(s,r)$ is an approximate function of $J(s)$;	θ is a discount rate for cost incurred in the future;
r is a vector of functional parameters;	$g(.)$ is a one-step cost function;
Δr is an adjustment to r;	$\phi(.)$ is a feature-extraction function;
u is a decision vector;	$e(.)$ is an error function.

2. Approximate Dynamic Programming

Even though DP provides exact solutions for optimisation over time, it suffers from the 'curses of dimensionality' (Powell, 2011). 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. In this section, we outline how ADP reduces this computational burden while extracting information from each optimisation to adapt to the operating environment and so to improve its decisions. Under the concept of reinforcement learning, certain specific techniques of temporal-difference learning are discussed, which form the basis of the present ADP framework.

2.1. Dynamic programming

Let $s \in S$ be a vector of state variables of the system, $u \in U$ the vector of decision variables, and $g(.)$ a function that calculates the cost during a step from one optimisation to the next based on the state and decision. Given an initial state s_t and future-discount factors α_i $(i \geq 0)$ a dynamic programme over a horizon of T steps calculates a sequence of decisions u from time step t by solving:

$$\min_{u\in U} E\left\{\sum_{i=0}^{T}\alpha_i g\left(s_{t+i}, u_{t+i}, s_{t+i+1}\right) | s_0\right\}. \tag{1}$$

The optimal performance J stating from state s_t at step t can be calculated by solving Bellman's recursive equation (Bellman, 1957):

$$J\left(s_t\right) = \min_{u\in U} E\left\{g\left(s_t, u, s_{t+1}\right) + \alpha J\left(s_{t+1}\left(u\right)\right)\right\}. \tag{2}$$

The backward dynamic programming procedure addresses this by starting from the final time T, assuming the optimal value function $J(s_{t+T})$ is known, and solving equation (2) recursively working backward to the initial state s_t to obtain the optimal sequence of decisions which would lead to the associated estimate of performance.

Although equation (2) characterises an elegant way of representing an optimisation problem, it can be computationally expensive and therefore is not practical for operational use. The reason for this is the need to consider the entire state space to calculate the optimal decision at each step, so that the computational intensity grows exponentially with additional state spaces.

Employment 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 time step may not be efficient, as plans for future decisions might be varied in light of emergence of new information so that decisions planned in the 'tail' period may never be implemented. Furthermore, complete information on future states might not be available owing to the stochastic environment. Although the opportunity will often arise to revise these decisions as their time approaches, calculating them in the first place is not necessarily a productive use of computational effort.

2.2. ADP with linear function approximation

In DP, the Bellman equation requires that at each step *t*, for each state in space *S* with *I* dimensions, decision in space *U* with *K* dimensions, and outcome in space *O* with *W* dimensions, a value function *J* be calculated to make an optimal decision. Therefore, the computational demand would increase quickly and exponentially as $S^I \times U^K \times O^W$. To address difficulties the consequent difficulties associated 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 future denoted as $g(.)$ together with an approximation $\hat{J}(.)$ to the value function $J(.)$ after that. This means that the ADP does not compute the exact value function $J(.)$, and so avoids the need to solve equation (2) for each state s_t at every future time step. Therefore, we reduce the future extent of the state space considered by replacing the true value function $J(s)$ with an approximation $\hat{J}(s, r)$ with parameters r but does not represent 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.

In this forward process, the focus is on the near-future for which information is more reliable and decisions imminent. Reinforcement learning is then used to update the parameters r of the approximation function $\hat{J}(s, r)$ according to the estimated optimal performance after each optimisation. At each time step t we estimate the value of the current state with explicit decisions in the near future and using the approximation $\hat{J}(.)$ to represent performance after that, implicitly assuming optimal decisions after time $t+T$. Thus

$$J\left(s_t\right) = \min_{u\in U} E\left\{\sum_{i=0}^{T}\alpha_i\ g\left(s_{t+i}, u_{t+i}, s_{t+i+1}\right) + \alpha_{t+T+1} J\left(s_{t+T+1}, r_t\right)\right\} \tag{3}$$

hence implementing at each time step t:

$$u_t^* = \arg\min_{u \in U} E\left\{ \sum_{i=0}^{T} \alpha_i \, g\left(s_{t+i}, u_{t+i}, s_{t+i+1}\right) + \alpha_{t+T+1} J\left(s_{t+T+1}, r_t\right) \right\} \tag{4}$$

Equation (3) calculates costs $g(.)$ only for states s_t that are visited, so reducing the computational burden substantially. These explicit cost calculations are moderated by use of the approximation function $\hat{J}(.)$ to represent future costs implicitly. This is one way in which ADP gains efficiency over backward dynamic programming. We adopt a separable linear approximation function for the approximate value function $\hat{J}(.)$, which can therefore be expressed as:

$$J\left(s, r\right) = r' \cdot \phi\left(s\right), \tag{5}$$

where $r = (r_1, r_2, \ldots, r_M)'$ is a column vector with each entry a parameter of the approximation function, and $\phi(.)$ is a features extraction function (or basis function) defined on the state space S and so maps the state to a feature vector for which r is the associated parameter vector and M is the number of features used. Tsitsiklis and Van Roy (1997) proved that for linear functions $\hat{J}(.)$, under certain assumptions, this approximation process converges to the unique optimal parameter vector r^*.

There are several different methods for updating the parameter vector r (Powell, 2011). In the present work, we explore some variants of the temporal difference (TD) learning techniques (Bradtke and Barto, 1996; Sutton, 1988). This reduces the difference between the approximated value function $\hat{J}(s, r)$ and the optimised value function $\tilde{J}(s)$, to improve the quality of approximations as more state transitions are observed. In the remainder of this section we build explore the literature on TD learning and introduce the implementations that were developed for the present application.

2.3. Temporal Difference learning

The approach of TD learning is to adjust the parameter vector r, once after each optimisation, to improve the estimated value by using the observed value. This is achieved by comparing the value $\hat{J}(s_t, r_t)$ for the current state s_t with the optimised value $\tilde{J}(s_t)$ from (3), to construct the measure of discrepancy:

$$\delta_t\left(r_t\right) = J\left(s_t\right) - J\left(s_t, r_t\right) \tag{6}$$

Combining equations (3) and (6), a temporal difference δ_t at time step t is described as:

$$\delta_t\left(r_t\right) = \sum_{i=0}^{T} \alpha_i \, g\left(s_{t+i}, u_{t+i}^*, s_{t+i+1}\right) + \alpha_{T+1} J\left(s_{t+T+1}, r_t\right) - J\left(s_t, r_t\right) \tag{7}$$

In this expression , the future discount factors $\alpha < 1$ are required because the future estimate $\hat{J}_{t+T}$ is augmented by the short-term values g_{t+i} before comparison with the immediate estimate $\hat{J}_t$. Standard TD literature adopts the quadratic error measure $e_t(r) = \delta_t^2(r)$ and calculates the M parameter values that would minimise this when aggregated over past time steps by solving after each step (4), the auxiliary optimisation

$$\underset{r}{\text{minimise}} \;\; E_t\left(r\right) = \sum_{i=0}^{t} \beta_i \, e_{t-i}\left(r\right) \tag{8}$$

for some positive discounting factors β_i $(0 \le i \le t)$ 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.

The solution to (8) can be used to calculate adjustments Δr_t to r_t that would minimise the current aggregated discrepancy as:

$$\Delta r_t = \arg\min_{\Delta r} \; E\left(r_t + \Delta r\right). \tag{9}$$

For the solution to (9) 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 (9) to (8).

In the case that $\beta_{M-1} = 0$, 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 (Bertsekas, 2011; Wagner, 2014), resulting in poor performance. We experienced this phenomenon while implementing TD for railway traffic management. Consequently, when using only current observations (*ie* $\beta_1 = 0$), we adopt moderated adjustments $r_{t+1} = r_t + \eta_t \Delta r_t$ where $\eta_t \in (0, 1)$ $(t \ge 1)$ is a decreasing sequence of moderating factors that satisfy the following conditions for convergence of r:

$$\sum_{t=1}^{\infty} \eta_t = \infty \qquad\qquad \sum_{t=1}^{\infty} \eta_t^{\,2} < \infty \tag{10}$$

In this approach, use of rapidly decreasing discounting factors β_i $(0 \le i \le t)$ that emphasise recent discrepancies will lead to greater adaptiveness of the ADP optimisation (3) 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.

For the cases where $\beta_{M-1} > 0$, the TD method applies the following Newton-based adjustment once after each optimisation:

$$\Delta r_t' = -\left(\sum_{j=0}^{t} \beta_j \nabla \delta_{t-j}\left(r_t\right)' \nabla \delta_{t-j}\left(r_t\right) \right)^{-1} \sum_{i=0}^{t} \beta_i \delta_{t-i}\left(r_t\right) \nabla \delta_{t-i}\left(r_t\right) \; . \tag{11}$$

Using the linear form of equation (5) for the approximate value function $\hat{J}(s, r)$, equation (7) becomes:

$$\delta_t = \sum_{i=0}^{T} \alpha_i \; g\left(s_{t+i}, u_{t+i}^{*}, s_{t+i+1}\right) + \alpha_{T+1} r_t' \cdot \phi\left(s_{t+T+1}\right) - r_t' \cdot \phi\left(s_t\right). \tag{12}$$

The optimisation (8) is then quadratic in r and is solved exactly, where it can be calculated, by (11) which becomes

$$\Delta r_t = -\mathbf{H}_t^{-1} \sum_{i=0}^{t} \beta_i \delta_{t-i}\left(r_t\right) \Phi_{t-i} \tag{13}$$

$$\text{where } \mathbf{H}_t = \sum_{j=0}^{t} \beta_j \Phi_{t-j} \Phi_{t-j}' \text{ and } \Phi_k = \alpha_{T+1} \phi\left(s_{k+T+1}\right) - \phi\left(s_k\right) \; \left(0 \le k \le t\right).$$

The initial value r_0 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.

One-step TD learning: $\beta_1 = 0$. 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 $-\nabla_r \delta_t = -(\alpha_{T+1}\phi_{t+T+1} - \phi_t)'$ 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:

$$\Delta r_t = -\mathbf{H}_t^{-1} \delta_t(r_t) \Phi_t \quad \text{where} \quad \mathbf{H}_t = \frac{1}{t} \sum_{j=0}^{t} \Phi_{t-j} \Phi_{t-j}' . \tag{14}$$

It can also be solved by using the Moore-Penrose pseudo-inverse $\mathbf{P}_t$ of the second derivative $\Phi_t \Phi_t'$ of e_t, thus $\Delta r_t = -\mathbf{P}_t \delta_t \Phi_t$, which will give another solution to $\delta_t(r_t + \Delta r_t) = 0$.

***M*-step TD learning**: $\beta_{M-1} > 0$, $\beta_M = 0$. In this case, provided that the information in the M discrepancies δ_i $(0 \le i < M)$ is mutually independent, then the matrix $\nabla_r^2 \delta_t$ of second derivatives will be invertible and there will be a unique solution r to (8) that achieves the global minimum of 0.

Least squares TD learning: $\beta_i = 1/t$ $(0 \le i \le t)$. This formulation (LSTD) was introduced by Bradtke and Barto (1996). After each optimisation (3), LSTD solves for parameters r by minimising the sum of squares of all temporal differences δ_i $(0 \le i \le t)$ since the start of operation. In this case, the objective

$$E_t(r) = \frac{1}{t} \sum_{i=0}^{t} \delta_{t-i}^2(r) . \tag{15}$$

of the optimisation (8) is quadratic in r and when $t \ge M - 1$ can be solved in a single Newton step. At this solution,

$$\nabla_r E_t(r_t) = 0' \quad (t \ge M - 1). \tag{16}$$

Thus for $t \ge M$, the solution to the minimization (8) is achieved by the single Newton step specified by (13). But from (15),

$$E_t(r) = \left(\frac{t-1}{t}\right) E_{t-1}(r) + \frac{1}{t} \delta_t^2(r) \quad (t \ge M). \tag{17}$$

$$\text{so that} \quad \Delta r_t = -\mathbf{H}_t^{-1} \left(\left(\frac{t-1}{t}\right) \nabla_r E_{t-1}(r_t) + \frac{1}{t} \delta_t(r_t) \Phi_t' \right) = \frac{-1}{t} \mathbf{H}_t^{-1} \delta_t(r_t) \Phi_t' \quad (t \ge M) . \tag{18}$$

This is a proportion $1/t$ of the one-step TD change given by (14), so with diminishing influence of subsequent discrepancies δ_t as t increases.

By incorporating all temporal differences up to step t, LSTD exploits at each step all the data observed since the start of the optimisation process. The weight given to the new information in the discrepancy δ_t at each step t as against past information, represented by the modification factors η_t, varies among the strategies according to the

choice of the discount parameters β_i $(0 \leq i \leq t)$. This is achieved at the computational expense of inverting the Hessian matrix $\mathbf{H}_t$ in equation (18).

3. Adaptive railway traffic control

We now present the application of ADP for railway traffic control. We discuss the control environment and the state space of the RTM system, as used for optimisation, then set out the algorithmic procedure that solves our RTM problem using ADP by employing the techniques derived in the previous section. Thus the control method proposed here can adapt according to prevailing traffic conditions on the railway network and calculate decisions accordingly.

3.1. System dynamics

Railway traffic is currently regulated using '*fixed block colour light signalling*', which divides the track into a series of fixed longitudinal sections called *blocks*. The operation of these systems is based on two main principles:

- A train cannot be authorised to enter a block if either it is currently occupied by another train, or one has already been authorised to enter it, and
- The distance separating a train from the next one downstream must always be greater than the braking distance required for the following train to stop safely.

Under these systems, the trains are regulated by controlling the colour signal lights positioned to the side or above the track, with a green signal indicating movement authority into the next block and a red signal indicating that the next block downstream is occupied. In case of three and four aspect railway signalling, yellow signals indicate braking for trains as they travel towards a red signal further downstream.

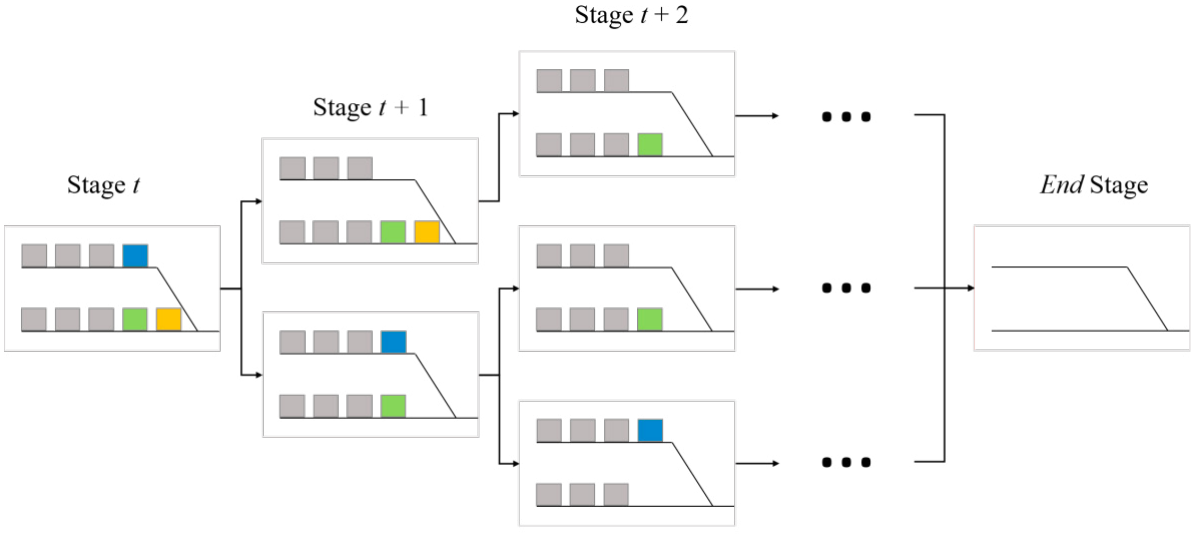


Fig. 1 – Representation of the state-space for a horizon of three trains in ADP.

In this study, we present a method for sequencing decisions at railway junctions that is intended to minimise the total consecutive delays of all trains. This objective function can be modified readily to include other considerations such as inclusion of importance weights to prioritise certain trains. In the present study we adopt a variant of the state space model for RTM presented by Ho et al. (1997) in which conflict resolution is treated as a multistage process in which each stage allows one train to pass through the junction and is characterised by the trains that then remain in the control area.

Excluding the initial stage, the system should undergo as many stages to the future as the number of trains currently in the control area: where this is extensive, the number can be substantial. The possible transformations for three trains are shown in Fig. 1. When many trains are present in the control area:, the number of possible states at the intermediate stages corresponding to Fig. 1 increases exponentially, making DP impractical as a solution method.

3.2. Control algorithm

We consider sequence controls at the time each train enters the designated control area: each such event corresponds to a stage of the dynamic optimisation. We therefore apply the general representation of ADP presented in the previous section to the adaptive control of railway traffic control, by considering decisions u_t at stage t over horizon of T stages to be considered explicitly. We calculate a sequence of decisions u_t^* by solving:

$$u_t^* = \arg\min_{u \in U} E\left\{\left[\sum_{i=0}^{T} \alpha_i g\left(s_{t+i}, u_{t+i}, s_{t+i+1}\right)\right] + \alpha_{T+1} J\left(s_{t+T+1}, r_t\right)\right\} \quad \left(1 \le t \le T\right) \tag{19}$$

where each stage cost $g\left(s_j, u_j, s_{j+1}\right)$ $(j \ge 0)$ represents the consecutive delay to all trains under consideration within the control area. This is repeated for a sequence of T entering trains. In the present case of railway traffic control the state s_t is a combination of traffic state and control state at stage t *ie* information on the trains remaining in the control area and awaiting movement authority, and the controls u specify the sequence of right-of-way assignments which optimises the junction capacity according to *g(.)*. The objective function therefore represents the stage cost functions $g(s_j, u_j, s_{j+1})$ of equation (19) together with the implicit future costs $\hat{J}(.)$. The resulting optimised value of the objective function is the used through equation (8) to update the approximation of the long-term total consecutive delays $\hat{J}(.)$. We adopt the rolling approach of implementing only the first of the calculated decisions (*ie* u_t^*) before recalculating (19) at the next stage.

In this framework, a temporal difference δ_t at stage t is:

$$\delta_t = \left[\sum_{i=0}^{T} \alpha_i g\left(s_{t+i}, u_{t+i}, s_{t+i+1}\right)\right] + \alpha_{T+1} J\left(s_{t+T+1}, r_t\right) - J\left(s_t, r_t\right). \tag{20}$$

We adopt a linear form (5) for the approximation function $J\left(s, r\right)$. The procedures for TD learning to calculate adjustments Δr_n to r_n are then as described in the previous section.

We developed the explicit stage cost functions $g(s_i, u_j, s_{j+1})$ for use in equation (19) and embedded them into a separate high fidelity microscopic railway traffic simulator which is configured for railway operations controlled by two, three and four aspect signals. 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 conflicts, 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.

The microscopic simulator employs Lomonossoff's equation to represent train movements inside each block section for the specified infrastructure by taking into consideration train characteristics (*eg* tractive force available), 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 all block sections and stations, 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. Thus, all safety and operational requirements are respected, and infrastructure data and the movement of trains is represented in detail. The temporal resolution adopted for this simulator is 0.1 second.

Here, we suppose that sufficient real-time data are available for railway systems to anticipate future arrival times into the control area. We also suppose that other data are available that describe the probability of events occurring in operations, *eg* distributions for dwell times at stations. Furthermore, we suppose that the latency of the system to receive data, calculate plans and deliver optimised sequences to the signaling system is short, *ie* the communication and computation delays are small so that control decisions can be calculated and implemented promptly.

Subject to this, the adaptive railway traffic control using ADP is summarised in Algorithm 1. In this formulation, the movement of the next T trains is represented explicitly, features ϕ based on the arrival times into the control area are extracted, and the optimal decision u_t^* are then calculated using (19). This requires evaluation of performance for u_t^*, which is then used to update approximation function by comparing the approximation $\hat{J}(s_t, r_t)$ to the optimised performance $\tilde{J}(.)$. The framework then implements the first part of the calculated plan, *ie* u_1^* or the first train to be given movement authority into the junction, and consider the next T trains, *ie t+1* to *T+1*. The framework then iterates through all remaining trains to produce the complete sequence of trains.

Algorithm 1 - ADP algorithm for adaptive railway traffic control

Step 1. Initialisation:
1.1 Choose an initial state s_0;
1.2 Initialise parameter vector r_0;
1.3 Choose the horizon T stages to be calculated explicitly;
1.4 Set $t = 0$;
1.5 Initiate learning rate η_0 (where required).
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 decisions u_t^* using (19).
Step 4. Update approximation function $\tilde{J}(.)$:
4.1 Calculate new observation $\tilde{J}(s_t)$ using (3);
4.2 Calculated current approximation $\hat{J}(s_t, r_t)$ using (5);
4.3 Calculate the T stage temporal difference δ_t at stage at the current stage t using (20);
4.4 TD learning:
Update parameter vector r_t to $r_{t+1} = r_t + \Delta r_t$ where Δr_t is calculated according to the TD learning strategy adopted *ie* (14) or (18).
Step 5. Implement decision:
5.1 Implement u_1^* ;
5.2 Update the state from s_t to $s_{t+1}(s_t, u_t)$ using the simulation
5.3 Return to step 2 if trains remain in the system, otherwise stop.

4. Numerical experiments and discussion

To evaluate our ADP approach using TD and LSTD learning techniques, infrastructure and operational data for a section of United Kingdom's railway was used to conduct numerical experiments. In this section, we first introduce the case study network 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 stochastic environment in which our experiments are conducted. Finally, our results of employing ADP for railway traffic control are presented and discussed. In the remainder of this paper we refer to our ADP approaches solely according to their learning techniques *ie* TD and LSTD.

4.1. 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 Hatfield and Stevenage, runs for approximately 20 kilometres, and includes five stations: Stevenage (major station), Knebworth, Welwyn North, Welwyn Garden City and Hatfield. Fig. 2 presents a detailed schematic of the infrastructure layout of the considered network which is electrified along its whole length at 25 kV AC and runs on conventional 4-aspect signalling.

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 that 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 and commuter services with frequent stops must negotiate usage of the same tracks. The problem of conflicting requests for use of tracks is exacerbated by commuter trains with planned stops at Welwyn North station, which block the line while dwelling at this station. Therefore, efficient management of railway traffic on this stretch is of paramount importance to the ECML, as it represents the primary bottleneck on the rail link connecting the eastern side of the UK from London and the South East of England to Scotland and the North.

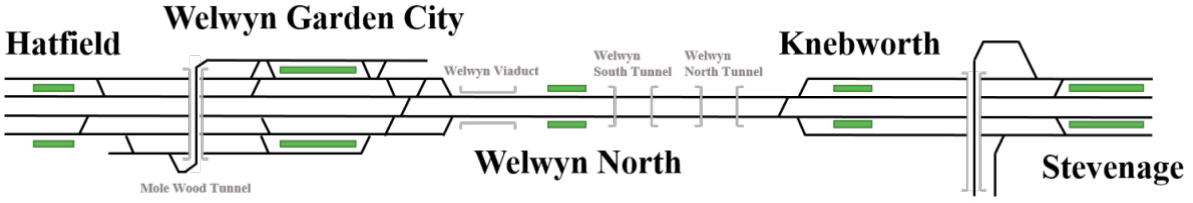


Fig. 2 – Infrastructure layout of the considered network

The 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 the present investigation, and so is ideal for our numerical experiments.

In our investigation, we simulate the 2018 timetable for a weekday between 7:00 am and 10:45 am. Northbound and southbound services do not interact on this network, so can be controlled independently. In this paper, we focus on sequencing trains at the bottleneck in Fig. 2 for southbound services (25 High-Speed Intercity services and 5 Commuter services with frequent stops) that run from Stevenage towards Hatfield and so consider the morning peak period on ECML towards London.

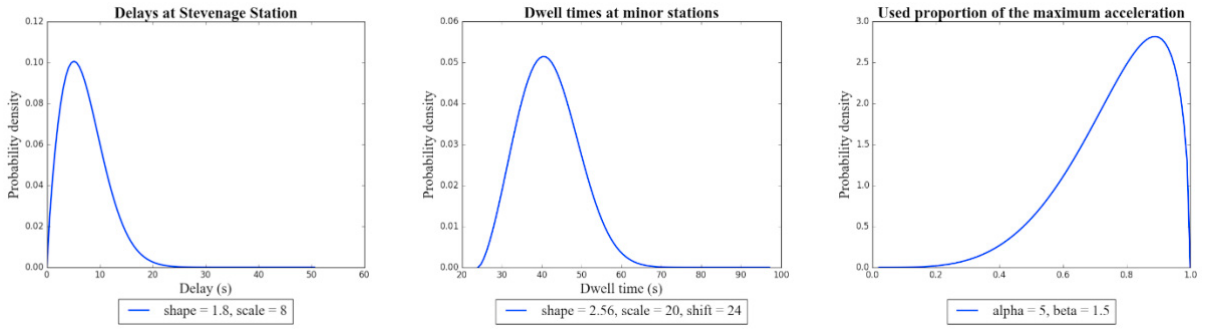
We perturb our timetable by delaying all trains that dwell at Stevenage Station. This includes all commuter services with planned stops at Welwyn North Station, as well as some intercity services. Half of the trains are perturbed by sampling from a Weibull distribution as Quaglietta et al. (2013) found to be appropriate for this: we adopt shape parameter 1.8 and scale parameter 8s, producing mean delays of 7.1 seconds (Fig. 3a). The individual delays can then be scaled to generate different traffic scenarios. Of 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, we evaluate each traffic scenario 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: Our 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, 20s, 24s as shape, scale and shift parameters respectively. This

distribution is reported by Quaglietta et al. (2013) for dwell times at minor stations in the Netherlands and is presented in Fig. 3b. The minimum dwell time at a minor station in our 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, we use a Beta distribution with shape parameters α and β as 5 and 1.5 respectively, shown in Fig. 3c, 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 robustness of this ADP formulation.



(a) (b) (c)

Fig. 3 – Probability distributions of (a) delays at Stevenage Station without scale factors, (b) dwell times at minor stations, and (c) the proportion of the maximum acceleration used by a driver on a block section.

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.

The discount factors α_n for use in (3) and (4) were calculated as $\alpha_n = e^{(-n\theta)}$, with the discount rate set at $\theta = 0.15$ (*ie* a discount of 15% for each successive train). An issue in using one-step TD learning is the choice of moderating step size η_t in updating the parameters r_t. A common approach in stochastic approximation is to use a stage-varying rate: we adopted a constant value for η_t so corresponding to the exponentially weighted variant.

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. The computational time for TD and LSTD are similar. This depends on the choice of T in equation (19) 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. In this study, we set $T = 3$ which translates into a computational time, in the stochastic environment, of 1.1 seconds per stage on average. Setting T to 4, 5 and 6 trains was found to increase the computational time to 2.86, 6.03 and 11.78 seconds per stage on average, respectively. This is not a specific limitation of ADP and arises from the exponential number of explicit calculations for the short-term costs $g(.)$ during in the stochastic environment.

4.2. Feature selection

Among various choices to represent the cost g in the objective function of the ADP (1) we considered minimising total consecutive delays. There are numerous possibilities for the features ϕ in equation (5) 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 $\hat{J}(.)$, not all combinations will 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, we simulated a traffic scenario 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) scheduled running time of remaining 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.

Fig. 4 presents the mean consecutive delay of all trains when they reach Hatfield Station (Fig. 2) for all 15 combinations of these features plus performance for 1 controller that uses no features, *ie* $M = 0$ and therefore always $\hat{J} = 0$; the FCFS performance for the traffic scenario was 20.48 seconds and is not shown on the graph as it performs substantially worse than any of the points shown in Fig. 4. Most combinations of features perform similarly to or worse than none at all (corresponding to a T step greedy heuristic without any approximation function $\hat{J}(.)$): the three combinations {a, c}, {a, b, c} and {a, b, d} did improve performance.

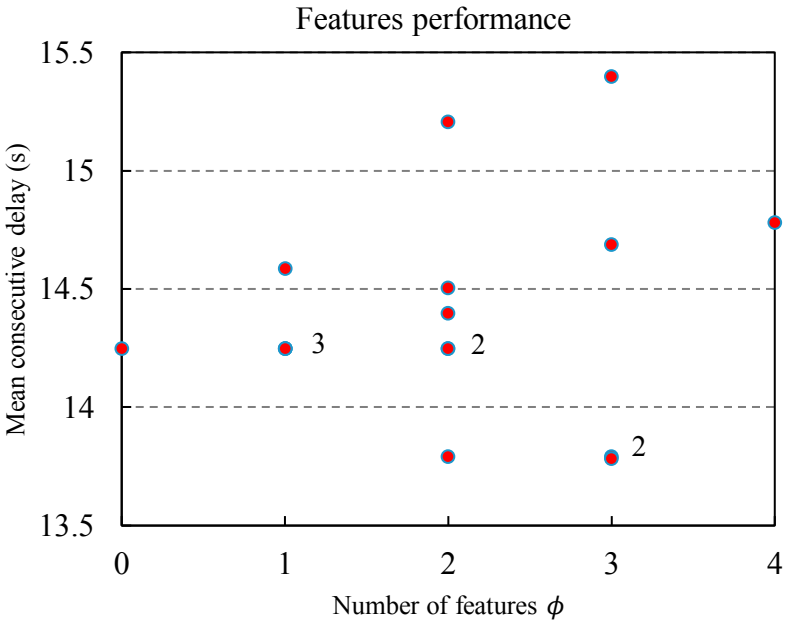


Fig. 4 – Performance of various feature combinations (multiple coincident values are indicated by number)

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. We adopted the combination {a, c} 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, which is clearly greater than that of the pure greedy algorithm that uses no features at all. 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. In the next section, we investigate in detail the performance of our 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 $\hat{J}(.)$.

4.3. Performance analysis

Two traffic scenarios were considered. For each, in total 30,000 train delays were generated from the Weibull distribution for Stevenage Station, equating to 2,000 morning peaks. Mean initial delay for all trains as measured just after Stevenage Station are 4 minutes 37 seconds for *Traffic Scenario A*, and 14 minutes 58 seconds for *Traffic Scenario B*. These compare with the mean timetable headway between all services on the shared section of Fig. 2 which is 4 minutes and 39 seconds. Accordingly, we aim to investigate ADP performance during moderate perturbations in *traffic scenario A*, and instances with more substantial perturbations in *traffic scenario B*.

In this section we adopt all of the adjustment Δr for LSTD (*ie* $\beta_l = 1/t$, $\eta_t = 1$) and, to compare traffic scenarios A and B we only adopt a tenth of the adjustment Δr for TD (*ie* $\beta_1 = 0$, $\eta_t = 0.1$). Towards the end of this section, a version of TD that adopts all adjustments of Δr (*ie* $\beta_1 = 0$, $\eta_t = 1.0$) is also considered.

Table 1 - Comparison of performance (seconds)

	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 1 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. These show that the proposed ADP approach reduces mean consecutive delay of FCFS by about a quarter in these scenarios. The two formulations of LSTD and one-step TD achieve almost the same performance, meaning the controls produced using the two learning methods are similar (*ie* they computed the same sequences in almost all cases). This need not mean they calculate the same approximations; indeed, the mean absolute error in approximation of value function $\hat{J}$ 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%.

Table 2 – Percentiles of consecutive delay (seconds)

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

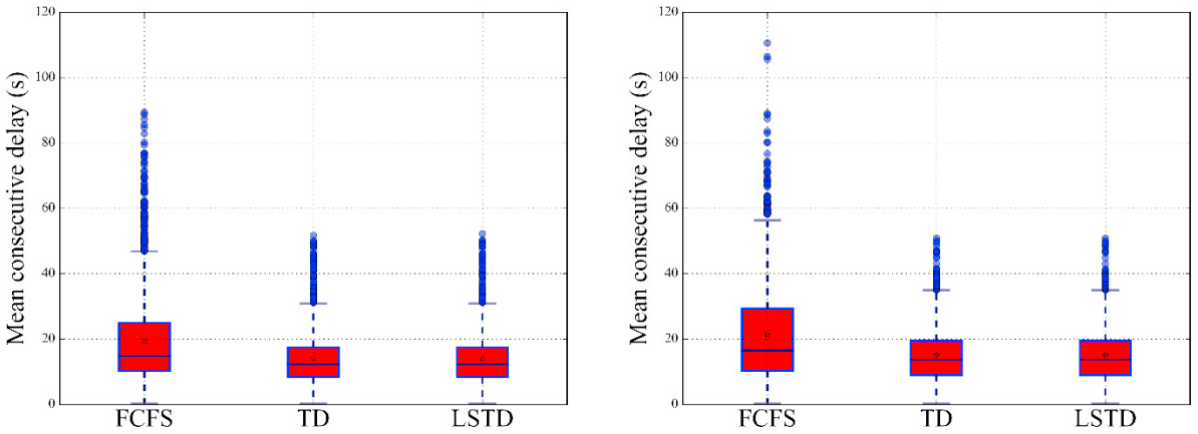
As the primary measure for our objective function, mean consecutive delays reveal how one-step TD and LSTD contributed towards their principal goal. One interesting observation is the similarity of the mean consecutive delays in traffic scenarios A and B. It seems that initial delays do not influence strongly the consecutive delays incurred inside the control area; this can also be observed in Fig. 5a and b, which present boxplots for the mean consecutive delays.

The main improvements in the high values of delay in Fig. 5 can be seen in the 75th and the 95th percentiles which are shown in Table 2. It is evident that the two learning techniques are similarly effective in reducing the proportion of long consecutive delays compared to FCFS. Of the two ADP methods, LSTD performs marginally better than does TD.

Table 3 – 75th Percentiles of train running times (seconds)

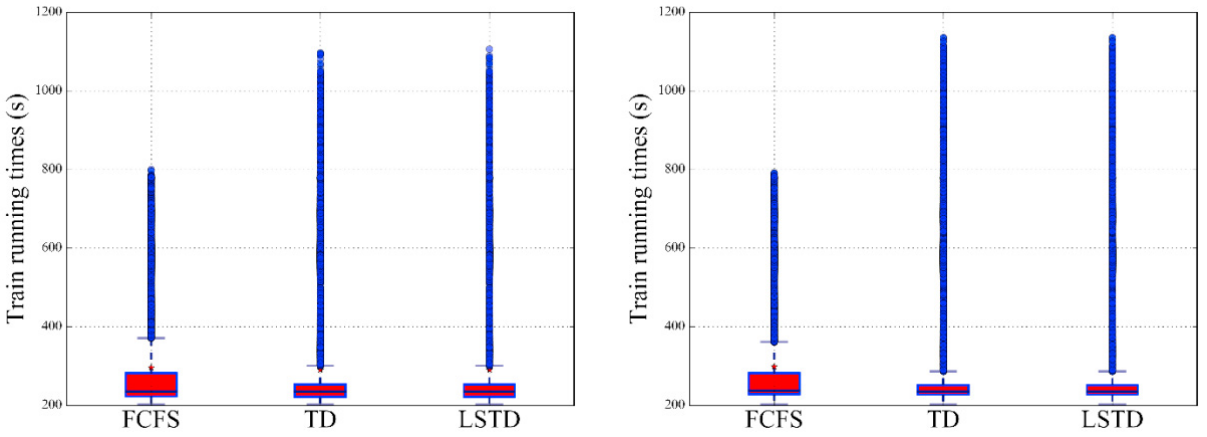
Traffic Scenario A			Traffic Scenario B		
FCFS	TD	LSTD	FCFS	TD	LSTD
281.92	253.31	253.31	281.69	251.24	251.25

As an indication of the rate of track occupancy within our test network, we study the running times of individual trains inside the control area. Fig. 6 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 75th percentile of Fig. 6, as is shown in Table 3. Unlike the mean consecutive delays, there is a long tail in TD and LSTD individual train running times of Fig. 6 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. The upper quartile (75th percentiles) in Table 3 are smaller than the corresponding mean values in Table 1 for running times, as a consequence of a small proportion of high values that are shown in Fig. 6.



(a) Scenario A (b) Scenario B

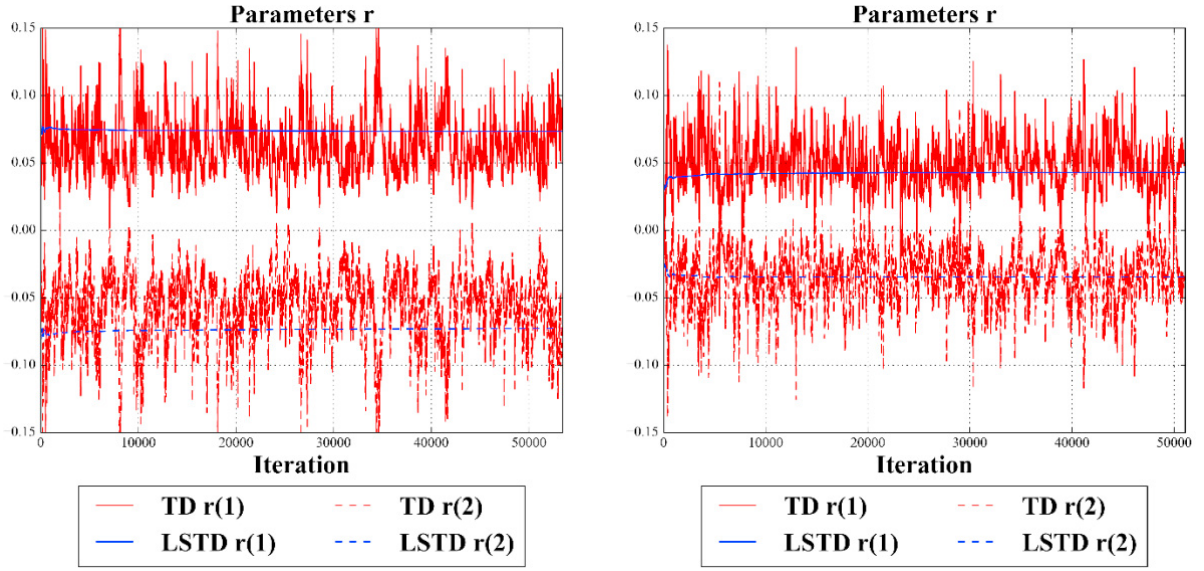
Fig. 5 – Distribution of mean consecutive delays.



(a) Scenario A (b) Scenario B

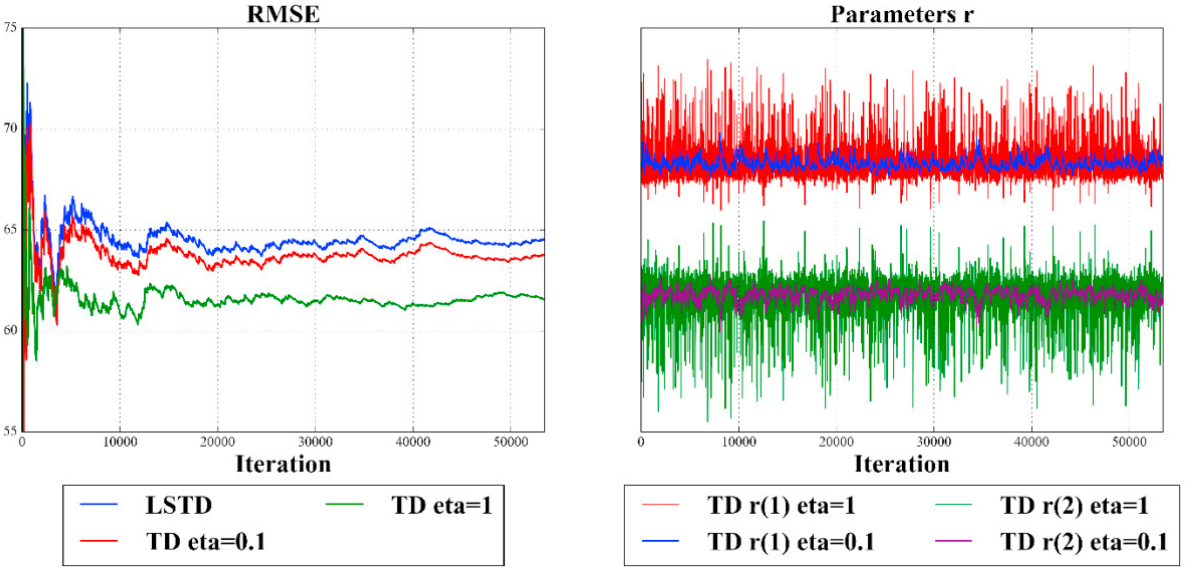
Fig. 6 – Distribution of individual train running times inside the control area.

Once more, in Fig. 6, 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 from performances in Fig. 4.



(a) Scenario A (b) Scenario B

Fig. 7 – Evolution and comparison of learning parameters for TD η_t of 0.1 and LSTD learning.



(a) (b)

Fig. 8 – (a) Root Mean Square Error of TD with η_t of 0.1 and 1, as well as LSTD learnings for traffic scenario A, and (b) evolution and comparison of learning parameters for TD with η_t of 0.1 and 1 for traffic scenario A.

The improvements in total delays in Table 1 are relatively small. Because of the ways that railways operate, most 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. Comparison of Figs 6a and 6b 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.

Fig. 7 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 Fig. 8(a), TD achieves smaller errors than does LSTD which suggest that TD tends to approximate the value function more accurately. The reason for this may be the nature of one step TD in reacting to short-term changes in traffic on the network whereas LSTD stays stable and approximate according to what has been observed from the start of the operations. Furthermore, Fig. 8(a) shows that $\eta_t = 1$ achieves 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 stability in the parameters as is clear in Fig. 8(b). 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.

Table 4 – 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

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 with similar potential for benefits in terms of operational performance. Tests using the exponentially weighted least squares TD learning approach with $\gamma \in \{0.5, 0.3, 0.1\}$ gave performance values that are almost indistinguishable from the LSTD approach so are not reported in detail here, though greater values of γ resulted in greater fluctuations in estimates of the parameters r of the approximation function.

5. Conclusions

This study presents an investigation into an adaptive railway traffic controller for real-time operations that applies approximate dynamic programming (ADP). 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 in the literature have investigated methods that are appropriate for use in stochastic environments. Many suffer from high computational complexity, which makes them inefficient or difficult to adopt for practical operation. We therefore have 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. We have formulated our problem as a dynamic program and use ADP to approximate the optimised value function, and reinforcement learning 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, we explore the variants of temporal difference (TD) and least-squares temporal difference (LSTD) learning techniques. Because number of features 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 present investigation found 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.

The numerical tests reported here 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 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.

Comparing the TD and LSTD approaches to estimation of parameters in the value approximation function, we found similar values between these approaches, 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 caluclations. 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. We also found that adopting all adjustments to parameters r 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 represented by $g(.)$ in equation (19) was found to be the most computationally expensive part of the present ADP framework. While testing ADP in a deterministic environment we achieved computational times of 0.07 second per stage on average for a lookahead horizon of $T = 3$ trains, 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 according to (3). This contrasts with the computational time required for FCFS, which was 0.016 second per stage in our experiments.

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 here. Therefore, the present ADP framework can be readily extended to include this capability. As an adaptive ADP approach for Railway Traffic Management (RTM), this study has so far 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.

Acknowledgements

This study is in part funded by the UK's Rail Safety and Standards Board (RSSB). The authors are grateful to colleagues at the Birmingham Centre for Railway Research and Education, University of Birmingham, UK for providing the authors with railway operational data for the Welwyn Viaduct area.

References

Adenso-Diaz, B., Oliva-Gonzalez, M., Gonzalez-Torre, P., 1999. On-line timetable re-scheduling in regional train services. Transp. Res. Part B Methodol. 33, 387–398.

Bellman, R., 1957. Dynamic Programming. Princeton University Press, Princeton.

Bertsekas, D.P., 2011. Approximate policy iteration : a survey and some new methods. J. Control Theory Appl. 9, 310–335. https://doi.org/10.1007/s11768-011-1005-3

Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T., 2017. Recent success stories on integrated optimization of railway systems. Transp. Res. Part C Emerg. Technol. 74, 196–211. https://doi.org/10.1016/j.trc.2016.11.015

Bradtke, S.J., Barto, A.G., 1996. Linear Least-Squares algorithms for temporal difference learning. Mach. Learn. 22, 33–57. https://doi.org/10.1007/BF00114723

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J., 2014. An overview of recovery models and algorithms for real-time railway rescheduling. Transp. Res. Part B Methodol. 63, 15–37. https://doi.org/10.1016/j.trb.2014.01.009

Cai, C., Wong, C.K., Heydecker, B.G., 2009. Adaptive traffic signal control using approximate dynamic programming. Transp. Res. Part C Emerg. Technol. 17, 456–474. https://doi.org/10.1016/j.trc.2009.04.005

Corman, F., Ariano, A.D., Pacciarelli, D., Pranzo, M., 2014. Dispatching and coordination in multi-area railway traffic management. Comput. Oper. Res. 44, 146–160. https://doi.org/10.1016/j.cor.2013.11.011

Corman, F., Ariano, A.D., Pacciarelli, D., Pranzo, M., 2012. Bi-objective conflict detection and resolution in railway traffic management. Transp. Res. Part C 20, 79–94. https://doi.org/10.1016/j.trc.2010.09.009

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010a. A tabu search algorithm for rerouting trains during rail operations. Transp. Res. Part B Methodol. 44, 175–192. https://doi.org/10.1016/j.trb.2009.05.004

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010b. Centralized versus distributed systems to reschedule trains in two dispatching areas. Public Transp. 2, 219–247. https://doi.org/10.1007/s12469-010-0032-7

Corman, F., Goverde, R.M.P., D'Ariano, A., 2009. Rescheduling Dense Train Traffic over Complex Station Interlocking Areas, in: Ahuja, R.K., Möhring, R.H., Zaroliagis, C.D. (Eds.), Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 369–386. https://doi.org/10.1007/978-3-642-05465-5_16

Corman, F., Meng, L., 2015. A Review of Online Dynamic Models and Algorithms for Railway Traffic Management. IEEE Trans. Intell. Transp. Syst. 16, 1274–1284.

D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007a. A branch and bound algorithm for scheduling trains in a railway network. Eur. J. Oper. Res. 183, 643–657. https://doi.org/10.1016/j.ejor.2006.10.034

D'Ariano, A., Pranzo, M., Hansen, I.A.I.A., 2007b. Conflict Resolution and Train Speed Coordination for Solving Real-Time Timetable Perturbations. IEEE Trans. Intell. Transp. Syst. 8, 208–222. https://doi.org/10.1109/TITS.2006.888605

Department for Transport, 2007a. Delivering a Sustainable Railway. London, United Kingdom.

Department for Transport, 2007b. Rail Technical Strategy 2007. London, United Kingdom.

Fang, W., Yang, S., Yao, X., 2015. A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks. IEEE Trans. Intell. Transp. Syst. 16, 2997–3016. https://doi.org/10.1109/TITS.2015.2446985

Ho, T.K., Norton, J.P., Goodman, C.J., 1997. Optimal traffic control at railway junctions, in: Electric Power Applications. IEE, pp. 140–148. https://doi.org/10.1049/ip-epa:19970941

Larsen, R., Pranzo, M., D'Ariano, A., Gorman, F., Pacciarelli, D., 2013. Susceptibility of optimal train schedules to stochastic disturbances of process times. Flex. Serv. Manuf. J. 26(4), 1–24.

Li, H., Womer, N.K., 2015. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. Eur. J. Oper. Res. 246, 20–33. https://doi.org/10.1016/j.ejor.2015.04.015

Mannino, C., Mascis, A., 2009. Optimal Real-Time Traffic Control in Metro Stations. Oper. Res. 57, 1026–1039. https://doi.org/10.1287/opre.1080.0642

Mascis, A., Pacciarelli, D., 2002. Job-shop scheduling with blocking and no-wait constraints. Eur. J. Oper. Res. 143, 498–517.

Mazzarello, M., Ottaviani, E., 2007. A traffic management system for real-time traffic optimisation in railways. Transp. Res. Part B Methodol. 41, 246–274. https://doi.org/10.1016/j.trb.2006.02.005

Medury, A., Madanat, S., 2013. Incorporating network considerations into pavement management systems : A case for approximate dynamic programming. Transp. Res. Part C Emerg. Technol. 33, 134–150. https://doi.org/10.1016/j.trc.2013.03.003

Mehta, F., Rößiger, C., Montigel, M., 2010. Latent energy savings due to the innovative use of advisory speeds to avoid occupation conflicts, in: Computers in Railways XII. WIT Transactions on The Built Environment, pp. 99–108. https://doi.org/10.2495/CR100

Meng, L., Zhou, X., 2011. Robust single-track train dispatching model under a dynamic and stochastic environment : A scenario-based rolling horizon solution approach. Transp. Res. Part B Methodol. 45, 1080–1102. https://doi.org/10.1016/j.trb.2011.05.001

Papadaki, K.P., Powell, W.B., 2003. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem, Naval Research Logistics. https://doi.org/10.1002/nav.10087

Papadaki, K.P., Powell, W.B., 2002. Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem. Eur. J. Oper. Res. 142, 108–127. https://doi.org/https://doi.org/10.1016/S0377-2217(01)00297-1

Papageorgiou, D.J., Cheon, M.-S., Nemhauser, G., Sokol, J., 2014. Approximate Dynamic Programming for a Class of Long-Horizon Maritime Inventory Routing Problems. Transp. Sci. 49, 870–885. https://doi.org/10.1287/trsc.2014.0542

Pellegrini, P., Marlière, G., Rodriguez, J., 2014. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. Transp. Res. Part B Methodol. 59, 58–80. https://doi.org/10.1016/j.trb.2013.10.013

Powell, W.B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality. John Wiley & Sons, Inc., Hoboken, New Jersey.

Powell, W.B., Bouzaiene-ayari, B., 2007. Approximate dynamic programming for rail operations, in: Mesa, C.L. and R.K.A. and J.A. (Ed.), 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'07). Dagstuhl, Germany, pp. 191–208. https://doi.org/10.4230/OASIcs.ATMOS.2007.1180

Quaglietta, E., Corman, F., Goverde, R.M.P., 2013. Stability analysis of railway dispatching plans in a stochastic and dynamic environment. J. Rail Transp. Plan. Manag. 3, 137–149. https://doi.org/10.1016/j.jrtpm.2013.10.009

Šemrov, D., Marseti˘c, R., Žura, M., Todorovski, L., Srdic, A., 2016. Reinforcement learning approach for train rescheduling on a single-track railway. Transp. Res. Part B Methodol. 86, 250–267. https://doi.org/10.1016/j.trb.2016.01.004

Simão, H.P., Day, J., George, A.P., Gifford, T., Nienow, J., Powell, W.B., 2009. An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application. Transp. Sci. 43, 178–197. https://doi.org/http:// dx.doi.org/10.1287/trsc.1080.0238

Sutton, R.S., 1988. Learning to Predict by the Methods of Temporal Differences. Mach. Learn. 3, 9–44. https://doi.org/10.1023/A:1022633531479

Törnquist, J., 2012. Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. Transp. Res. Part C Emerg. Technol. 20, 62–78. https://doi.org/10.1016/j.trc.2010.12.004

Törnquist, J., 2007. Railway traffic disturbance management — An experimental analysis of disturbance complexity , management objectives and limitations in planning horizon. Transp. Res. Part A Policy Pract. 41, 249–266. https://doi.org/10.1016/j.tra.2006.05.003

Törnquist, J., Persson, J.A., 2007. N -tracked railway traffic re-scheduling during disturbances Johanna To. Transp. Res. Part B Methodol. 41, 342–362. https://doi.org/10.1016/j.trb.2006.06.001

Tsitsiklis, J.N., Van Roy, B., 1997. An Analysis of Temporal-Difference Learning with Function Approximation. IEEE Trans. Automat. Contr. 42, 674–690. https://doi.org/10.1109/9.580874

Van Roy, B., Bertsekas, D.P., Lee, Y., Tsitsiklis, J.N., 1997. Neuro-Dynamic Programming Approach to Retailer Inventory Management, in: Proceedings of the 36th IEEE Conference on Decision and Control. IEEE, San Diego, CA, USA, USA. https://doi.org/10.1109/CDC.1997.652501

Wagner, P., 2014. Policy oscillation is overshooting. Neural Networks 52, 43–61. https://doi.org/10.1016/j.neunet.2014.01.002

Yin, J., Tang, T., Yang, L., Gao, Z., Ran, B., 2016. Energy-efficient metro train rescheduling with uncertain time-variant passenger demands : An approximate dynamic programming approach. Transp. Res. Part B Methodol. 91, 178–210. https://doi.org/10.1016/j.trb.2016.05.009

Zhang, W., Dietterich, T., 1995. A Reinforcement Learning Approach to Job-Shop Scheduling, in: IJCAI'95. Montreal, Canada, pp. 1114–1120.