# Distributed Approximate Dynamic Control for Traffic Management of Busy Railway Networks

Taha Ghasempour, Gemma L. Nicholson, David Kirkwood, Taku Fujiyama, and Benjamin Heydecker

***Abstract*— Railway operations are prone to disturbances that can rapidly propagate through large networks, causing delays and poor performance. Automated re-scheduling tools have shown the potential to limit such undesirable outcomes. This study presents the network-wide effects of local deployment of an adaptive traffic controller for real-time operations that is built on approximate dynamic programming (ADP). The controller aims to limit train delays by advantageously controlling the sequencing of trains at critical locations. By using an approximation to the optimised value function of dynamic programming that is updated by reinforcement learning techniques, ADP reduces the computational burden substantially. This framework has been established for isolated local control, so here we investigate the effects of distributed deployment. Our ADP controller is interfaced with a microscopic railway traffic simulator to evaluate its effect on a large and dynamic railway system, which controls critical points independently. The proposed approach achieved a reduction in train delays by comparison with First-Come-First-Served control. We also found the improvements to be greater at terminal stations compared to the vicinity of our control areas.**

***Index Terms*— Approximate dynamic programming, railway traffic management, adaptive control, reinforcement learning.**

## I. Introduction

RAILWAY networks around the world, in recent decades, have become complicated, with a mixture of single track, double track and multitrack sections, running several different classes of trains with different operational characteristics such as speeds. The complexity has come about because of the sustained increase in passenger numbers where in many countries railways are increasingly popular as means of commuting to cities.

In the UK, the increase in passenger flows has put substantial pressure on the existing rail network, and as a result, train frequencies are close to network capacity on some lines. To address these challenges the government and railway undertakings are encouraging development of intelligent transport systems to regulate and optimise train operations in real-time to increase railway capacity and customer satisfaction while at the same time decreasing cost and energy usage [1], [2]. The reason behind this drive is to add more useable capacity and reliability into the railway network by improved usage of current infrastructure and capabilities at affordable cost.

Real-time regulation of railway traffic aims at ensuring safe, punctual and energy-efficient train operations. Tools, which are known collectively as Railway Traffic Management (RTM) systems, anticipate future traffic conflicts based on current train dynamic status information and provide suitable control measures by using appropriate mathematical models. By measuring the position and speed of trains in real-time and assessing requirements and opportunities, the motion of trains can be controlled advantageously through real-time management of train sequencing and of acceleration and deceleration of individual trains. This facilitates the response to minor deviations from scheduled operation and to major disruptions to expedite recovery. In doing so it will underwrite the operation of enhanced timetables that meet the increasing call on rail network capacity by passenger and freight operations.

A growing body of literature is available on real-time RTM [3], [4] which has proved and demonstrated theoretically the effectiveness and benefits of these tools. Among the reported formulations for RTM, Mixed Integer Linear Programming (MILP) and Alternative Graph (AG) are among the most widely described approaches.

AG is a discrete optimisation approach which can be used to model re-scheduling problems with no-wait and no-store constraints which can be applied to job shops [5]. Several works are reported in the literature using the AG model to formulate railway re-scheduling; among the first, D'Ariano *et al.* [6] proposed a branch-and-bound 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. This formulation allows for presentation of railway network safety constraints so that at any time, only one train is present in any block section. This algorithm is designed to minimise the greatest duration of paths in the AG corresponding to minimising the maximum consecutive delay. Consecutive delays are defined as delays that are propagated to other trains from the initial delays that arise in traffic, e.g. longer than planned dwell times. Building on their previous

Manuscript received August 21, 2018; revised March 24, 2019 and June 29, 2019; accepted July 25, 2019. This work was supported in part by the U.K.'s Rail Safety and Standards Board (RSSB). The Associate Editor for this article was R. Goverde. *(Corresponding author: Taha Ghasempour.)*

T. Ghasempour, T. Fujiyama, and B. Heydecker are with the Centre for Transport Studies, Faculty of Engineering Science, University College London, London WC1E 6BT, U.K. (e-mail: taha.ghasempour.14@ucl.ac.uk; taku.fujiyama@ucl.ac.uk; b.heydecker@ucl.ac.uk).

G. L. Nicholson and D. Kirkwood are with the Birmingham Centre for Railway Research and Education, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: g.l.nicholson@bham.ac.uk; d.kirkwood@bham.ac.uk).

Digital Object Identifier 10.1109/TITS.2019.2934083

work, D'Ariano *et al.* [7] adopted a blocking time model to evaluate the feasibility of headways between successive trains, and train speed profiles are updated considering preceding signal aspects. Mazzarello and Ottaviani [8] 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.

Corman *et al.* [9]–[12] extended the work of D'Ariano *et al.* [6] to include local re-routing of trains with the objective of minimising the maximum consecutive delay. Corman *et al.* [13], [14] extend their work by introducing a supervisory layer to their controller that coordinates multiple local areas to control large networks. For a limited number of areas, the framework is reported to perform well, but as the prediction horizon is extended and the number of local areas is increased, good solutions could not be guaranteed. Re-scheduling techniques based on AG have been shown to perform well under stochastic disturbances to operations [15].

Törnquist and Persson [16] 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 computational times, Törnquist [17], [18] extends their work by presenting a heuristic approach to solve the same problem, and reported that good solutions could be produced fast. Another instance of MILP formulation for re-scheduling was formulated by Pellegrini *et al.* [19] 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 considered trains. A rolling horizon approach was adopted for evaluation to examine different horizon periods. More recently, Samà *et al.* [20] developed metaheuristic algorithms based on MILP for re-scheduling and re-routing of trains in complex and busy railway networks.

A scan of the available literature on RTM reveals that comparatively few prototypes of these systems have been implemented in practice [21]–[23]. One significant difficulty with the approaches to real-time train re-scheduling is that they are heavily affected by the size of problem instances and the issue of computational difficulty of solving the RTM problem has been encountered by many of the studies reviewed in this section. The nature of rail operations means that for congested and complex railway networks obtaining optimal solutions could be computationally intractable due to the number of cases that need to be considered.

A comprehensive study of implementing re-scheduling approaches for large and dynamic networks has been presented by Quaglietta *et al.* [24], where AG and MILP approaches are shown to be computationally efficient when tested on large networks. They demonstrate the importance of computationally efficient re-scheduling methods for real-time railway traffic management.

It is thus the aim of this paper to build upon the existing re-scheduling methods and develop an optimisation framework that uses a data-based approach to make decisions reliably and in a timely manner and so respect the practical requirements of railway operations. This will improve the prospect for practical implementation of real-time traffic management systems.

In this paper, we build on the work in [25] and investigate the distributed implementation of a local control method based on Approximate Dynamic Programming (ADP) for large and dynamic railway networks. ADP has been studied extensively and applied effectively in practical applications where large state and action spaces exist [26]–[32]. ADP is a variant of Dynamic Programming (DP) which uses an approximation to the optimal performance function to evaluate options. It is usually used when the robustness of a DP is needed but the problem is too complicated for a full DP strategy to be solved in an efficient and timely manner. 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 performance that will result from future states with optimal control and plans accordingly. Although the use of ADP to solve operational railway problems is rare, studies in railway related ADP are emerging [33]–[35].

The aim of this paper is to investigate the effectiveness of ADP in tackling railway re-scheduling for busy and dynamic networks. Given that the goal of the RTM system is to achieve network optimisation in a timely manner, we evaluate our framework in a realistic simulation environment in a distributed deployment across a large section of the East Coast Main Line railway in the UK to control known critical areas of the network and evaluate network-wide performance.

In the remainder of this paper we describe our traffic control framework which is based on ADP in section II. In section III we set out the procedure used to evaluate our traffic controller. Section IV introduces our test case network, discusses the distributed setup in which our ADP framework is tested and presents findings from our computational experiments. We conclude this paper and consider the scope for future research in section V.

## II. Control Framework

Real-time RTM involves making decisions based on known current status of the railway network, then observing in real-time the consequent status, after which further decisions are made. Such problems are known as sequential decision problems, where decisions have to be made for the anticipatable future, and subsequent decisions are made, once the system has developed and further information becomes available.

DP provides a convenient way to formulate problems of this kind. Even though DP provides exact solutions for optimisation over time, it suffers from the 'curses of dimensionality' [36]. 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. This makes DP impractical for real-time railway operations.

In this section, we outline how ADP reduces this computational burden while learning online from information to adapt and improve its decisions. Under the concept of reinforcement learning, least-squares temporal-difference learning (LSTD) is discussed, which forms the basis of the learning mechanism

in our ADP framework. Finally, at the end of this section we adopt ADP for our RTM problem.

### A. Dynamic Programming

Let $s \in S$ be the state variable of the system, $u \in U$ the decision variable, and $g$ the one step cost function. Given the initial state $s_0$ and a sequence of decisions $u_t$ at time step $t$, a future-discounted dynamic programme over a horizon of $T$ steps is:

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

where state $s_{i+1}$ depends on state $s_i$ and decision $u_i$. This can be solved recursively according to Bellman's equation [37]:

$$J\left(s_t\right) = \min_{u_t \in U} E\left\{g\left(s_t, u_t, s_{t+1}\right) + \alpha J\left(s_{t+1}\right)\right\}, \quad \text{for } 0 \le t \le T-1 \tag{2}$$

where $J$ is the value function, $\alpha$ is the discount factor that is defined by $\alpha_t = \exp\left(-t\theta\right)$ and $\theta$ is a discount rate for cost incurred in the future, and $E$ is the expectation over random information.

Although equation (2) can be used to calculate the sequence of decisions which would lead to the optimal solution, it can be computationally expensive and therefore is not practical for operational use. DP considers the entire state space to evaluate the optimal decision at a single step, and the computational intensity grows exponentially with additional state variables.

Employment of DP for real-time control is especially onerous. Considering the entire state space to the end of the planning period at every time step is likely to be inefficient, as increasingly many decisions will need to be evaluated at each step and will then need to be re-evaluated in the future as new information emerges, meaning decisions planned for the latter part of the period may not remain appropriate.

### B. Approximate Dynamic Programming

To address the main difficulties of solving DP, approximate dynamic programming has been developed to limit the extent of computation required at each step by focusing on the immediate future. This approach adopts a functional approximation $\hat{J}\left(s, r\right)$ with parameters $r$ for the future performance of the system from state $s$ under optimal control. This approximation is used as part of an exhaustive search of feasible sequences of decisions over a limited horizon $T$. An explicit model of the system is used to estimate the development over the time steps $t$ up to $T$ from state $s_t$ under controls $u_t$ to state $s_{t+1}(s_t, u_t)$ with associated performance $g(s_t, u_t)$. The accumulated performance during the explicit evaluation horizon associated with each sequence is then supplemented by an estimate of future costs $\hat{J}\left(s_T, r\right)$ provided by the approximation function. The sequence of decisions that leads to the best estimate of performance is then implemented, either in part or in whole. This approach includes explicit evaluation for the exhaustive search of the short-term future followed by approximate evaluation of the state remaining at the end of that period: it therefore limits the evaluation and calculation of control to states in the short-term future for which information tends to be more accurate and decisions more relevant. The computational requirement is therefore polynomial to the number of state variables rather than exponential to the size of the state space.

In this forward process, the more reliable part of the information is used to make decisions, and, in our case, reinforcement learning is then used to update the approximations according to result of each optimisation. At each time step $t$ we estimate the future performance under optimal control commencing from the current state:

$$\tilde{J}\left(s_t\right) = \min_{u_t \in U} E\left\{g\left(s_t, u_t, s_{t+1}\right) + \alpha \hat{J}\left(s_{t+1}, r_t\right)\right\}, \quad \text{for } t = 0, 1, \ldots, T-1 \tag{3}$$

Hence implement at each time step $t$:

$$u_t^* = arg \min_{u_t \in U} E\left\{g\left(s_t, u_t, s_{t+1}\right) + \alpha \hat{J}\left(s_{t+1}, r_t\right)\right\}. \tag{4}$$

Equation (3) is only calculated if state $s_t$ is visited, consequently reducing the computational burden significantly. This is precisely how ADP gains efficiency over DP. A lookup table could be used to store the value function associated with state $s_t$, however, the goal of ADP is to increase computational efficiency, therefore parameterised value function approximators are used to store the value function more compactly. We employ a separable linear approximation function to express the approximate value function $\hat{J}\left(.\right)$, which can be expressed as:

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

where $r = \left(r\left(1\right), r\left(2\right), \ldots, r(M)\right)'$ 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 weight vector and $M$ is the number of features used.

To approximate the optimal value function using equation (5) requires a learning process to update the parameter vector $r$. This is termed 'learning' in the ADP community and lies at the heart of any ADP approach.

There are several different methods for updating the parameter vector $r$ [36], among them Reinforcement Learning techniques, and, in particular, temporal-difference learning introduced by Sutton [38], have received considerable attention in the literature and have been adopted successfully in ADP approaches for adaptive traffic management [39].

In this investigation, we explore the least-squares temporal difference learning technique [40] which is a method for approximating long-term future cost as a function of current state. The fundamental idea behind LSTD learning involves an incremental process that minimises the error between the approximated value function $\hat{J}\left(s, r\right)$ and the observed value function $\tilde{J}\left(s\right)$, with the objective of improving approximations as more state transitions are observed. We introduce the LSTD learning technique next.

### C. LSTD Learning

The goal of LSTD learning is to solve for parameter vector $r$, once after every observation, such that the state's predicted value $\hat{J}(s_t, r_t)$ better represents the state's observed value $\tilde{J}(s_t)$. Hence, using equation (3), a temporal difference is defined as:

$$\delta_t = g\left(s_t, u_t^*, s_{t+1}\right) + \alpha \hat{J}(s_{t+1}, r_t) - \hat{J}(s_t, r_t). \quad (6)$$

After each observation, LSTD solves for parameters $r$ by minimising the sum of squares of all temporal differences since the start of operation:

$$e(r) = \frac{1}{2}\sum_{j=1}^{t} \delta_j^2, \quad (7)$$

$$r = arg\ \min_{r} \{e(r)\}, \quad (8)$$

by taking the partial derivative of equation (7) with respect to $r$:

$$\frac{\partial e}{\partial r} = \sum_{j=1}^{t} \delta_j \nabla_r \delta_j. \quad (9)$$

Using equations (5) and (6) to expand equation (9), parameters $r$ can be found by:

$$r = A_t^{-1} b_t, \quad (10)$$

where $A_t$ and $b_t$ are:

$$A_t = \sum_{j=1}^{t} \left(\phi(s_j) - \alpha_{j+1}\phi(s_{j+1})\right).\left(\phi(s_j) - \alpha_{j+1}\phi(s_{j+1})\right)', \quad (11)$$

$$b_t = \sum_{j=1}^{t} \left(\phi(s_j) - \alpha_{j+1}\phi(s_{j+1})\right) g\left(s_j, u_j^*, s_{j+1}\right). \quad (12)$$

Derivation of equation (10) can be found in the Appendix. To implement LSTD for real-time control, we incorporate each observed cost and state transition into matrix $A$ and vector $b$, then solve for an updated parameter vector $r$. It should be noted that once $A$ and $b$ are updated, the experience can be overlooked without losing information.

### D. ADP for Railway Traffic Control

The specific objective of this study is to provide sequencing decisions at railway junctions that minimise the total running times of services inside a control area, calculated as the difference between control area exit and entry times. It is worth noting that the objective function can be changed to include other considerations such as prioritising certain trains, but the parameter training process would need to be re-initiated in such cases. In this study we adopt a variant of the state-space model for RTM presented by Ho *et al.* [41], where 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 left in the control area. After the initial stage, the system should undergo as many stages into the future as the number of trains currently in the control area; where this is extensive, the number can be substantial. The possible transitions for three trains are shown in Fig. 1. When many trains are present in the control area: to, the number of possible states at the intermediate stages corresponding to

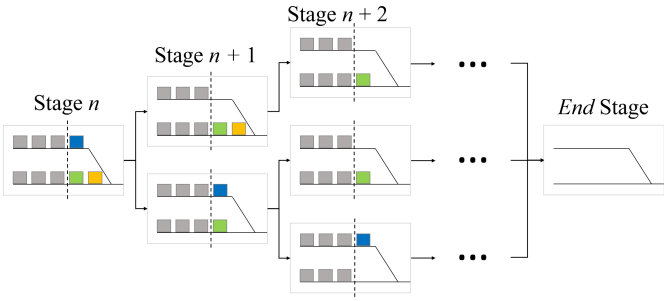



Fig. 1. Representation of the state-space for a horizon of three trains in ADP.

Fig. 1. increases exponentially, making DP impractical as a solution approach.

We consider sequence controls at the time each train enters a designated control area: each such event corresponds to a stage $n$ of the dynamic optimisation. We therefore distinguish between the general representation of ADP, presented in the previous section, and the ADP for railway traffic control, by considering decisions $u_n$ at stage $n$, over a horizon of $N$ stages to be considered explicitly, thus calculating a sequence of decisions $u_N^*$ using:

$$u_N^* = arg\ \min_{u_N \in U} E\left\{\left[\sum_{i=n}^{n+N} \alpha_{i-n} g(s_i, u_i, s_{i+1})\right] + \alpha_{N+1} \hat{J}(s_{n+N+1}, r_n)\right\}, \quad (13)$$

where each stage cost $g(s_i, u_i, s_{i+1})$ represents the running time of a train through the control area. In the present case of railway traffic control, the state $s_n$ comprises traffic state and control state at stage, i.e. 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 $g(s_i, u_i, s_{i+1})$ of equation (13) together with the estimated future costs $\hat{J}(.)$ in which optimal control is implicit. The resulting optimised value of the objective function is then used through equations (6) to (8) to update the approximation of the long-term performance $\hat{J}(.)$. We adopt the rolling approach of implementing only the first of the calculated decisions (i.e. $u_n^*$) and transfer to the next stage.

In this framework, a temporal difference $\delta_n$ at stage $n$ is:

$$\delta_n = \left[\sum_{i=n}^{n+N} \alpha_{i-n}\, g(s_i, u_i, s_{i+1})\right] + \alpha_{N+1} \hat{J}(s_{n+N+1}, r_n) - \hat{J}(s_n, r_n). \quad (14)$$

The LSTD procedure of equation (10) is then used to calculate functional parameters $r$ at each stage of optimisation.

To explicitly evaluate the $N$ stage cost function $g$ in equation (13), we have developed and embedded a microscopic railway traffic simulator into our ADP framework. This simulator is used to evaluate short-term performance of trains in the control area. To do so, Lomonossoff's equation [42] is used to formulate trains' motion for the specified infrastructure, train characteristics, signalling system, interlocking constraints,

and stations. At the heart of this microscopic traffic simulator is the calculation of continuous braking curves, which are calculated for trains to stop at the end of each block section or station, and for transitions into lower speed limits. Thus, all safety and operational requirements are respected.

This simulator is used to determine the complete feasible solution space at each stage by identifying and considering all sequences that could be realised, respecting, for example that on plain track a leading train cannot be overtaken by following trains. These feasible solutions are then evaluated explicitly by simulating interactions between all trains based on the decisions allowing for computation of exit times from the control area, which is beyond the conflict point, and thus calculate individual train delays for a horizon of $N$ trains. This is the one-step cost $g(.)$. The long-term approximation $\hat{J}(.)$ is then used to estimate trains' performance (i.e. total running times of all trains beyond $N$) also at the exit from the control area for trains that are planned to enter the system in the future. Therefore, decisions are made from a combination of short-term explicit evaluation and long-term estimates of performance.

There are numerous choices for features $\phi$ in equation (5) and the cost during a single stage $g$ in equation (13). By conducting experiments on various combinations of features $\phi$ and one-stage cost $g$ we found that many of these combinations lead to inferior performance. In this paper, we present results for a combination that was found to improve performance. For features $\phi$ in equation (5) we extract the scheduled running times of remaining trains in the control area ($\phi_1$) and the time differences between services at the conflict point according to the current plan ($\phi_2$) to approximate value functions $\hat{J}(s, r)$. Therefore, according to equation (5), $r_1$ and $r_2$ are the associated weights of $\phi_1$ and $\phi_2$ respectively, and the scalar product of the $\phi$ vector and the $r$ vector is the approximation $\hat{J}(s, r)$.

The ADP procedure adopted is as follow: the movement of the next $N$ trains is evaluated explicitly, features $\phi$ based on the arrival times into the control area are extracted, and the optimised decision $u_n^*$ is then calculated using (13). The evaluation of performance for $u_n^*$, is then used to update approximation function $\hat{J}(s_n, r_n)$ by comparing it to the optimised performance $\tilde{J}(s_n)$. 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 considers the next $N$ trains, i.e. $n+1$ to $N+1$. The framework then iterates through subsequent trains to produce the complete sequence of all trains.

## III. Evaluation Framework

A separate high-quality microscopic railway traffic simulator, BRaVE [43], is used to evaluate the performance of the interlocking and signalling across a model network; this is different to the microscopic simulator embedded in our ADP framework described in Section II.D. An Application Programming Interface (API) mechanism enables external actors to control the signalling and trajectories of trains within the pre-defined control areas of the model network. The API mechanism communicates XML format messages containing outgoing traffic state and incoming optimised sequences, i.e. the sequence of right-of-way assignments for each of the conflict points.

The ADP controller interfaces with the simulator via a network connection and communicates XML format messages. As a simulation runs, traffic state messages are periodically sent to the ADP controller, which monitors the state of the network and optionally sends back optimised sequencing instructions for the conflict points; in the current paper, traffic states are sent to the controller every time trains enter the control area or every 5 minutes, whichever is sooner. On receipt of these instructions the simulator implements them, just as a signalling decision is implemented at a junction, while the ADP controller continues to monitor the traffic state. This process continues until the simulation ends at a predefined time.

Here, we suppose that sufficient real-time data are available to anticipate future arrival times into the control area. We also suppose that dwell times and driver behaviours are deterministic and known. Furthermore, we suppose 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.

## IV. Computational Experiments

To investigate the effects resulting from distributed deployment of our ADP framework on a large and dynamic railway network, infrastructure and operational data for the United Kingdom's East Coast Main Line (ECML) was used to conduct computational experiments. In this section, we present our case study network as well as results and discussion of our experiments.

### A. 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. Fig. 2. 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.

We have identified two points on this network which are critical in terms of traffic flow on the ECML. 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.

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

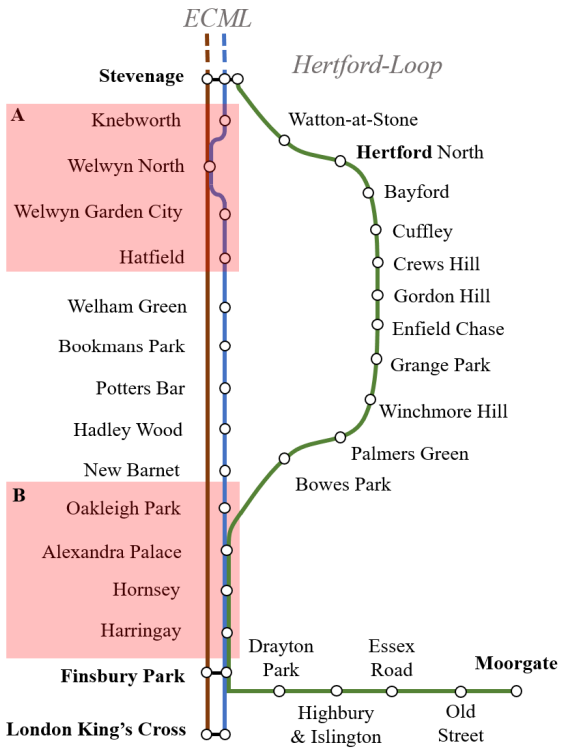



Fig. 2. Schematic of the study network, with red boxes indicating, A: Digswell control area, and B: Finsbury control area.

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. 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.

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.

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

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 part of the network; therefore, they can be controlled independently. In this paper, we focus on southbound services from Stevenage station, in total 42 services. This consists of 25 high-speed intercity services and 6 commuter services with frequent stops on the

TABLE I

TRAIN RUNNING TIMES (SECONDS) BY CONTROL AREA AND SERVICE GROUPING

| | 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 |

mainline of ECML and 11 commuter services on the Hertford Loop and in to London.

We perturb the traffic 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 our trains are perturbed by sampling from a Weibull distribution with shape parameter 1.8 and scale parameter 8 producing a mean delay of 7.1 seconds. 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 delayed services dwelling at Stevenage. Once trains leave Stevenage, running times and subsequent dwelling periods are treated deterministically.

In this study, the First-Come-First-Served (FCFS) heuristic provides the benchmark on the lower end of performance. No benchmark for the higher end is considered as finding optimal solutions to our RTM problem is computationally very expensive and not practically achievable for real-time operations.

The ADP framework is not trained in advance and the parameters $r$ are set to the value 0 at the beginning of each simulation run. All numerical experiments were conducted by computer simulation under Windows 10 on a PC configured with Intel Core i7- 4790 CPU and 32 GB RAM. For simulations of this paper, we achieved computational times of 0.07 second per stage $n$ on average.

### B. Results and Discussion

In total 28000 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.

In this section, we focus on train running times inside the control area, which is the primary objective of our controller, and on station delays as a measure of performance in the network as a whole.

Table I 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 the mean running time of all trains

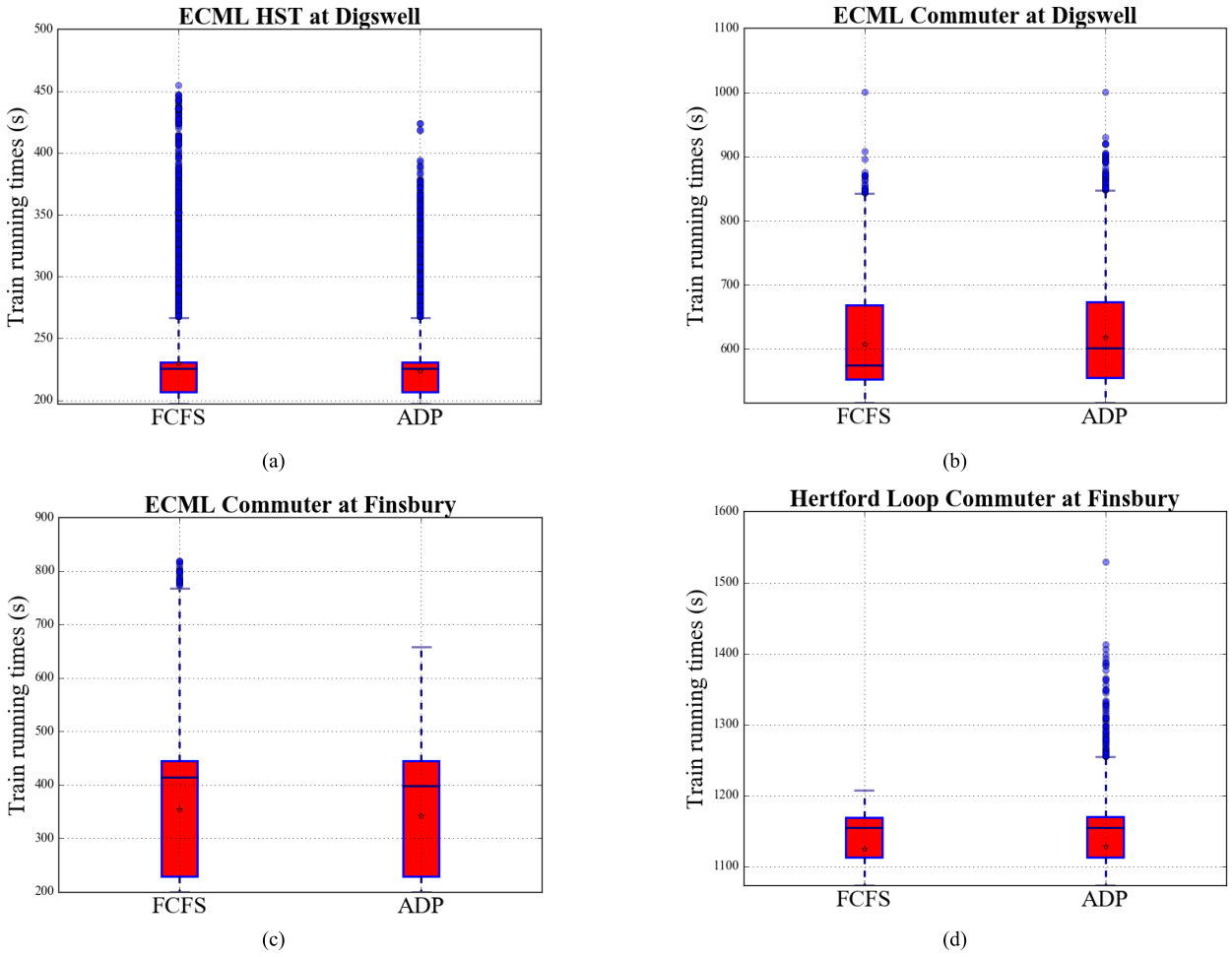



Fig. 3. Running times of services inside the control areas separated by service groupings; (a) ECML HST within Digswell control area, (b) ECML commuter services within Digswell control area, (c) ECML commuter services within Finsbury control area, and (d) Hertford-Loop commuter services within Finsbury control area; star signs indicate mean values.

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 an average of 11% and 21% reduction in added running times (*ie.* added delays) 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. In the majority of instances, this is achieved by allowing faster trains to overtake slower regional trains to avoid them being held behind them and so delayed.

To further analyse the effects of optimisation on the performance of services, we break down the running times into service groupings. When comparing running times inside Digswell control area in Fig. 3a and Fig. 3b and also in Table I, it is clear that ECML HST services benefit from the re-scheduling, whereas ECML commuter services are disadvantaged. This trend is repeated within Finsbury control area (Fig. 3c and d) where ECML commuter trains benefit from optimisation but 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. Because the control framework seeks to minimise the total running times of all trains, the effect of re-scheduling when taking into account all trains leads to faster trains being favoured in the optimised control plans. In the case of Digswell control area, the controller produces plans that reduce running times of the 25 ECML HST trains by an average of 6.46 seconds whilst increasing the running times of the 6 ECML commuter trains by an average of 10.79 seconds resulting in a total saving of 96.76 seconds. Similarly, for the Finsbury control area, the 6 ECML commuter trains achieve average improvements of 12.04 seconds while the 11 Hertford-Loop commuter trains see an increase of 3.13 seconds their mean running times inside the control area resulting in a total saving of 37.81 seconds.

Table II 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

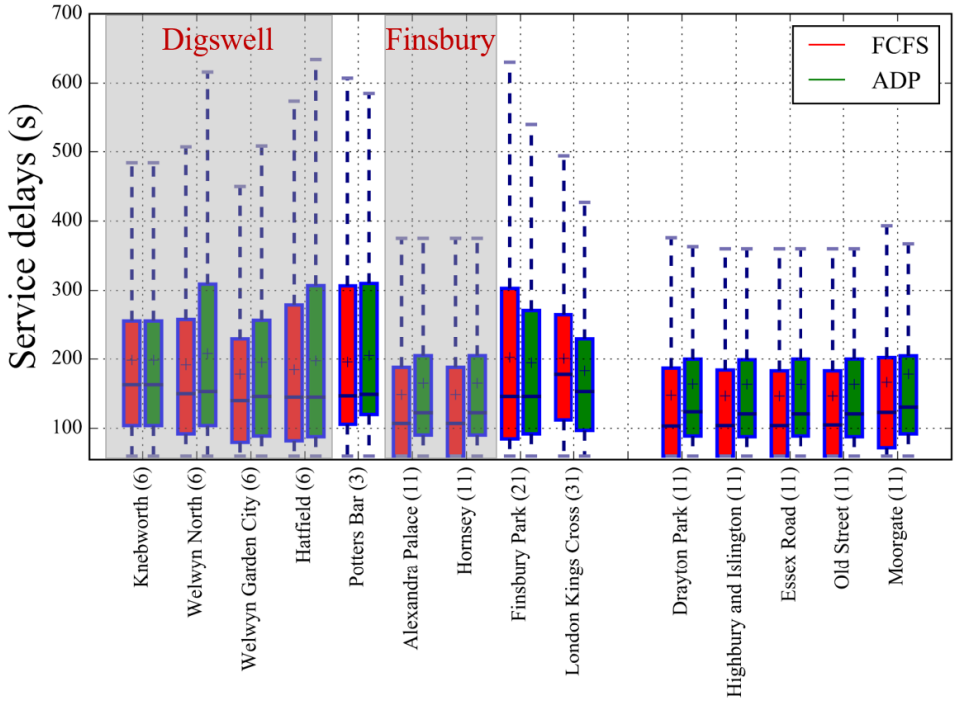



Fig. 4. Comparison of individual delays of over 1 minute between FCFS and ADP for all trains stopping at stations inside or downstream of the control areas; numbers in brackets indicate total number of scheduled stops per simulation at the station, and + signs indicate mean values.

TABLE II
LIKE BY LIKE MEAN DELAYS THAT EXCEED *X* SECONDS

| Major stations (N° 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 |

trains, and Finsbury Park station is where all ECML commuter trains and most ECML HST services stop (15 services in our case). The mean delays have been categorized by taking the mean value of delays of over 0, 1, 3 and 5 minutes. UK rail industry separates delays using these criteria 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. Mean delays in Table II are calculated for the subset of trains that experience delays in these ranges under each of FCFS or ADP control.

The trend observed in mean running times within control areas can also be observed in Table II, 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 achieved by ADP of 9.6%, 10.4%, 15.3% and 26.3% respectively 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) and Moorgate (11), it is clear that improvements have been achieved by applying ADP on this network.

Fig. 4 presents all arrival delays of over 1 minute for all stations within or downstream of our control areas. This shows effects at other stations within the network. The 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 Fig. 4 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.

Fig. 5 presents the evolution of functional parameters for LSTD at the Digswell and Finsbury control areas, truncated at 200 iterations. Recalling that LSTD learning solves for parameters $r$ by minimising the sum of squares of all temporal

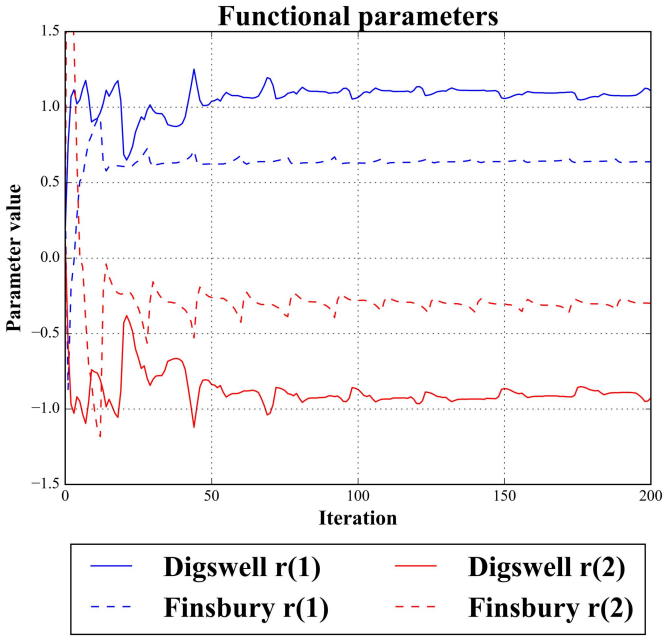



Fig. 5. Evolution of functional parameters $r$.

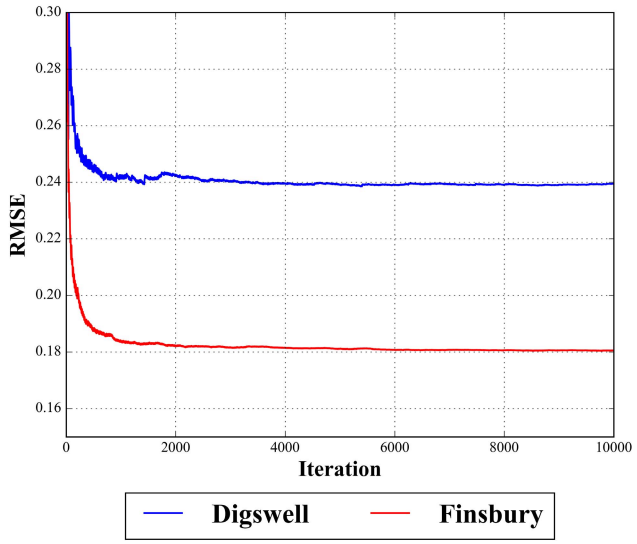



Fig. 6. Proportionate Root Mean Square (RMSE) $\delta_n$ (14) of LSTD for the two control areas.

difference errors since the start of operation, as expected the $r$ values stabilise early on and after few learning opportunities. The values for the functional parameters stabilise on different values for the two control areas, showing that the convergent values are specific to control area, timetable and other features of the operating environment. Fig. 6 shows the root mean square deviation $\delta_n$ (14) of the LSTD approximations $\hat{J}(s,r)$ expressed as a proportion of the estimated value, which fall and stabilise as information accumulates.

## V. Conclusion

This study presents network-wide effects of a local adaptive traffic controller for real-time operations that is built on approximate dynamic programming (ADP). The controller aims to limit train delays by controlling the sequencing of trains at critical locations. Many methods presented in the literature for railway traffic control suffer from high computational complexity which makes them inefficient or difficult to adopt for practical operations. We therefore have employed the ADP framework to reduce the computational burden. This algorithmic framework substantially reduces the number of states to be evaluated, which leads to a corresponding reduction in the computational burden. We have formulated our problem as a dynamic programme, adopting ADP to approximate the optimal value function with reinforcement learning techniques (LSTD) to update the approximation. We deployed our framework in a realistic simulation environment in a distributed setup across a large and dynamic network and controlled known critical areas of the network and evaluated network-wide performance.

Given that the penalty system for delayed trains in the UK is mainly concerned with delays at large or terminal station we focused our analysis on the major stations within our study network. We found improvements in service delays at the largest station within our study network (London King's Cross) which was the terminal station for most of the services that we considered, 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 we considered delays of all trains at their terminal stations. We also note that over the 3¾hour period of our scenarios, the controller 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 $3^{1/2}$ minutes, it may be worth investigating whether implementing a re-scheduler similar to that presented here could facilitate addition of a further HST service in the ECML timetable each day.

As for the ADP framework, the RMSE of functional parameters stabilise after about 1000 learning opportunities; in practice they could be calculated according to a period of training using historical data then implemented. Given this observation, the full parameters training would take around 70 simulation seconds per junction in our case. Care is required in selecting features as we found some to be detrimental to performance compared to a greedy heuristic. We found that minimising total running times of services resulted in improved performance on our study network, though further research would be required to test and compare other objective functions for ADP and on other networks of similar or higher complexity.

Because the most computationally intensive part of the present ADP framework is the explicit evaluation of the one-step cost $g(.)$ in equation (13), investigation of other, perhaps macroscopic, approaches and their effect on train performance needs to be considered.

The framework presented here uses data generated locally by trains to approximate performance of railway traffic in real-time and so to calculate signal controls. This does not

guarantee optimality of solutions but favours practicality above optimality. Little of the literature available on RTM considers data-based approaches to solve this complex optimisation. The distributed implementation of ADP presented in this paper is a systematic and self-tuning adaptive approach which we have shown to achieve better performance than FCFS by considering the consequence of decisions in a timely manner. It is therefore well 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. This study has focused on the distributed independent application of the control policy at isolated junctions operating independently of each other. The applicability of ADP to co-ordinate network traffic is a promising 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.

## APPENDIX

Here we present the derivation of the LSTD formulation of equation (10), which is as follows. Using equations (5) and (6) we have:

$$\delta_t = g\left(s_t, u_t^*, s_{t+1}\right) + \alpha_{t+1} r_t^{'} \phi\left(s_{t+1}\right) - r_t^{'} \phi\left(s_t\right), \quad (15)$$

and hence:

$$\nabla_r \delta_j = \alpha_{j+1} \phi\left(s_{j+1}\right) - \phi\left(s_j\right). \quad (16)$$

Therefore, by expanding equation (9) we find:

$$\begin{aligned} 0 &= \sum_{j=1}^{t} \left(\phi\left(s_j\right) - \alpha_{j+1}\phi(s_{j+1})\right)\left(g\left(s_j, u_j^*, s_{j+1}\right)\right. \\ &\quad \left. + \alpha_{j+1} r^{'} \phi\left(s_{j+1}\right) - r^{'}\phi(s_j)\right), \\ &= \sum_{j=1}^{t} \left(\phi\left(s_j\right) - \alpha_{j+1}\phi\left(s_{j+1}\right)\right) g\left(s_j, u_j^*, s_{j+1}\right) \\ &\quad - \sum_{j=1}^{t} \left(\phi\left(s_j\right) - \alpha_{j+1}\phi\left(s_{j+1}\right)\right).\left(\phi\left(s_j\right)\right. \\ &\quad \left. - \alpha_{j+1}\phi\left(s_{j+1}\right)\right)' r, \end{aligned} \quad (17)$$

and so the $r$ values are computed using:

$$0 = b_t - A_t r \quad (18)$$

## REFERENCES


[1] *Delivering a Sustainable Railway*, Dept. Transp., London, U.K., 2007.

[2] *Rail Technical Strategy 2007*, Dept. Transp., London, U.K., 2007.

[3] V. Cacchiani *et al.*, "An overview of recovery models and algorithms for real-time railway rescheduling," *Transp. Res. B, Methodol.*, vol. 63, pp. 15–37, May 2014.

[4] F. Corman and L. Meng, "A review of online dynamic models and algorithms for railway traffic management," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1274–1284, Jun. 2015.

[5] A. Mascis and D. Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints," *Eur. J. Oper. Res.*, vol. 143, no. 3, pp. 498–517, 2002.

[6] A. D'Ariano, D. Pacciarelli, and M. Pranzo, "A branch and bound algorithm for scheduling trains in a railway network," *Eur. J. Oper. Res.*, vol. 183, no. 2, pp. 643–657, Dec. 2007.

[7] A. D'Ariano, M. Pranzo, and I. A. Hansen, "Conflict resolution and train speed coordination for solving real-time timetable perturbations," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 208–222, Jun. 2007.

[8] M. Mazzarello and E. Ottaviani, "A traffic management system for real-time traffic optimisation in railways," *Transp. Res. B, Methodol.*, vol. 41, no. 2, pp. 246–274, Feb. 2007.

[9] F. Corman, R. M. P. Goverde, and A. D'Ariano, "Rescheduling dense train traffic over complex station interlocking areas," in *Robust and Online Large-Scale Optimization*. Berlin, Germany: Springer, 2009, pp. 369–386.

[10] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "A tabu search algorithm for rerouting trains during rail operations," *Transp. Res. B, Methodol.*, vol. 44, no. 1, pp. 175–192, Jan. 2010.

[11] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "Bi-objective conflict detection and resolution in railway traffic management," *Transp. Res. C, Emerg. Tech.*, vol. 20, no. 1, pp. 79–94, 2012.

[12] 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," *Transp. Planning Technol.*, vol. 34, no. 4, pp. 341–362, 2011.

[13] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "Centralized versus distributed systems to reschedule trains in two dispatching areas," *Public Transp.*, vol. 2, no. 3, pp. 219–247, 2010.

[14] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "Dispatching and coordination in multi-area railway traffic management," *Comput. Oper. Res.*, vol. 44, pp. 146–160, Apr. 2014.

[15] R. Larsen, M. Pranzo, A. D'Ariano, F. Corman, and D. Pacciarelli, "Susceptibility of optimal train schedules to stochastic disturbances of process times," *Flexible Services Manuf. J.*, vol. 26, no. 4, pp. 466–489, 2014.

[16] J. Törnquist and J. A. Persson, "N-tracked railway traffic re-scheduling during disturbances," *Transp. Res. B, Methodol.*, vol. 41, no. 3, pp. 342–362, Mar. 2007.

[17] J. Törnquist, "Railway traffic disturbance management—An experimental analysis of disturbance complexity, management objectives and limitations in planning horizon," *Transp. Res. A, Policy Pract.*, vol. 41, no. 3, pp. 249–266, Mar. 2007.

[18] J. T. Krasemann, "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances," *Transp. Res. C, Emerg. Technol.*, vol. 20, no. 1, pp. 62–78, Feb. 2012.

[19] P. Pellegrini, G. Marlière, and J. Rodriguez, "Optimal train routing and scheduling for managing traffic perturbations in complex junctions," *Transp. Res. B, Methodol.*, vol. 59, pp. 58–80, Jan. 2014.

[20] M. SamàÂ , A. D'Ariano, F. Corman, and D. Pacciarelli, "A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations," *Comput. Oper. Res.*, vol. 78, pp. 480–499, Feb. 2017.

[21] C. Mannino and A. Mascis, "Optimal real-time traffic control in metro stations," *Oper. Res.*, vol. 57, no. 4, pp. 1026–1039, 2009.

[22] F. Mehta, C. Rößiger, and M. Montigel, "Latent energy savings due to the innovative use of advisory speeds to avoid occupation conflicts," in *Proc. Int. Conf. Comput. Syst. Design Oper. Railway Transit Syst.*, 2010, pp. 99–108.

[23] R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther, and T. Schlechte, "Recent success stories on integrated optimization of railway systems," *Transp. Res. C, Emerg. Technol.*, vol. 74, pp. 196–211, Jan. 2017.

[24] E. Quaglietta *et al.*, "The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture," *Transp. Res. C, Emerg. Technol.*, vol. 63, pp. 23–50, Feb. 2016.

[25] T. Ghasempour and B. Heydecker, "Adaptive railway traffic control using approximate dynamic programming," *Transp. Res. C, Emerg. Technol.*, to be published.

[26] H. P. Simao, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell, "An approximate dynamic programming algorithm for large-scale fleet management: A case application," *Transp. Sci.*, vol. 43, no. 2, pp. 178–197, 2009.

[27] K. P. Papadaki and W. B. Powell, "Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem," *Eur. J. Oper. Res.*, vol. 142, no. 1, pp. 108–127, Oct. 2002.

[28] H. Li and N. K. Womer, "Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming," *Eur. J. Oper. Res.*, vol. 246, no. 1, pp. 20–33, Oct. 2015.

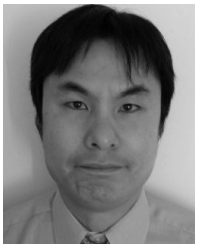
[29] W. Zhang and T. Dietterich, "A reinforcement learning approach to job-shop scheduling," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, Aug. 1995, pp. 1114–1120.

[30] A. Medury and S. Madanat, "Incorporating network considerations into pavement management systems: A case for approximate dynamic programming," *Transp. Res. C, Emerg. Technol.*, vol. 33, pp. 134–150, Aug. 2013.
[31] B. Van Roy, D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis, "A neuro-dynamic programming approach to retailer inventory management," in *Proc. 36th IEEE Conf. Decis. Control*, Dec. 1997, pp. 4052–4057.
[32] D. J. Papageorgiou, M.-S. Cheon, G. Nemhauser, and J. Sokol, "Approximate dynamic programming for a class of long-horizon maritime inventory routing problems," *Transp. Sci.*, vol. 49, no. 4, pp. 870–885, 2014.
[33] W. B. Powell and B. Bouzaiene-ayari, "Approximate dynamic programming for rail operations," in *Proc. 7th Workshop Algorithmic Methods Models Optim. Railways*, 2007, pp. 191–208.
[34] D. Šemrov, R. Marsetic, M. Žura, L. Todorovski, and A. Srdic, "Reinforcement learning approach for train rescheduling on a single-track railway," *Transp. Res. B, Methodol.*, vol. 86, pp. 250–267, Apr. 2016.
[35] J. Yin, T. Tang, L. Yang, Z. Gao, and B. Ran, "Energy-efficient metro train rescheduling with uncertain time-variant passenger demands: An approximate dynamic programming approach," *Transp. Res. B, Methodol.*, vol. 91, pp. 178–210, Sep. 2016.
[36] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ, USA: Wiley, 2011.
[37] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.
[38] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.
[39] C. Cai, C. K. Wong, and B. G. Heydecker, "Adaptive traffic signal control using approximate dynamic programming," *Transp. Res. C, Emerg. Technol.*, vol. 17, no. 5, pp. 456–474, Oct. 2009.
[40] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Mach. Learn.*, vol. 22, nos. 1–3, pp. 33–57, Jan. 1996.
[41] T. K. Ho, J. P. Norton, and C. J. Goodman, "Optimal traffic control at railway junctions," *IEE Proc.-Electr. Power Appl.*, vol. 144, no. 2, pp. 140–148, Mar. 1997.
[42] S. Lu, S. Hillmansen, and C. Roberts, "A power-management strategy for multiple-unit railroad vehicles," *IEEE Trans. Veh. Technol.*, vol. 60, no. 2, pp. 406–420, Feb. 2011.
[43] H. Douglas, P. Weston, D. Kirkwood, S. Hillmansen, and C. Roberts, "Method for validating the train motion equations used for passenger rail vehicle simulation," *Proc. Inst. Mech. Eng. F, J. Rail Rapid Transit*, vol. 231, no. 4, pp. 455–469, 2017.

**Taha Ghasempour** received the B.Eng. degree in mechanical engineering from City, University of London, London, U.K., in 2013. He is currently pursuing the Ph.D. degree in transport systems engineering with University College London (UCL), London, Prior to this, he worked in the U.K. rail industry as a Researcher. His current research concerns the application of ADP and reinforcement learning in intelligent railway control systems.

**Gemma L. Nicholson** received the MMath and M.Sc. degrees in mathematics from the University of Bath, Bath, U.K., in 2004 and 2005, respectively, and the Ph.D. degree for research into the design of microwave bandpass filters, in 2009. She is currently a Research Fellow at the Birmingham Centre for Railway Research and Education. Her research is in a range of applied problems using modeling, simulation, and signal processing techniques.

**David Kirkwood** received the B.Sc. degree in computing science from Staffordshire University, U.K., in 2002, and the Ph.D. degree in railway traffic control simulation from the University of Birmingham, U.K., in 2014. He has 15 years of software development experience in industry and academia, and has developed object-oriented simulators for air traffic control and for railways. He is currently a Research Fellow at the Birmingham Centre for Railway Research and Education, where he develops simulation software.



**Taku Fujiyama** received the B.Sc. degree in civil engineering from Waseda University, Japan, in 1995, and the Ph.D. degree in pedestrian modeling from the University College London (UCL). He worked at the East Japan Railway Company and was involved in infrastructure development projects, before coming to UCL. He is currently an Associate Professor with UCL, and focuses on railway traffic optimization and passenger movement.

**Benjamin Heydecker** received the B.A. degree in mathematics from the University of Cambridge, Cambridge, U.K., in 1978, and the Ph.D. degree in transport studies from UCL, in 1983. Following his postdoctoral research at ITS, University of Leeds, he was appointed as a Lecturer at UCL, where he is currently a Professor of transport studies. He applies mathematical and statistical analysis to transport systems, with special interest in emerging ICT.