Transportation Research Part B 140 (2020) 210–235

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

An actor-critic deep reinforcement learning approach for metro train scheduling with rolling stock circulation under stochastic demand

Cheng-shuo Ying[a], Andy H.F. Chow[b,*], Kwai-Sang Chin[a]

[a] Systems Engineering and Engineering Management, City University of Hong Kong, Kowloon Tong, Hong Kong SAR
[b] Architecture and Civil Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong SAR

ARTICLE INFO

Article history:
Received 8 January 2020
Revised 17 July 2020
Accepted 19 August 2020

Keywords:
Metro train scheduling
Stochastic transit demand
Actor-critic architecture
Deep reinforcement learning
Multi-objective optimization

ABSTRACT

This paper presents a novel actor-critic deep reinforcement learning approach for metro train scheduling with circulation of limited rolling stock. The scheduling problem is modeled as a Markov decision process driven by stochastic passenger demand. As in most dynamic optimization problems, the complexity of the scheduling process grows exponentially with the amount of states, decisions, and uncertainties involved. This study aims to address this 'curses of dimensionality' issue by adopting an actor-critic deep reinforcement learning solution framework. The framework simplifies the evaluation and searching process for potential optimal solutions by parameterizing the original state and decision spaces with the use of artificial neural networks. A deep deterministic policy gradient algorithm is developed for training the artificial neural networks via simulated system transitions before the actor-critic agent can be applied for online schedule control. The proposed approach is tested with a real-world scenario configured with data collected from the Victoria Line of London Underground, UK. Experiment results illustrate the advantages of the proposed method over a range of established meta-heuristics in terms of computing time, system efficiency, and robustness under different stochastic environments. This study innovates urban transit operations with state-of-the-art computer science and dynamic optimization techniques.

1. Introduction

Metro systems are vital for sustainable urban development with their high carrying capacity and eco-efficiency. Performances of metro operations are subject to a number of factors including capacity of rolling stock, availability of staff, operational protocols, and variability of passenger demand. Compared with the intercity mainline operations (see Cacchiani et al., 2016 and Chow and Pavlides, 2018), the unique challenges encountered in urban metro operations include the busy service schedules (up to 32–34 trains per hour), frequent circulation of rolling stock, unexpected perturbations, and the dynamic and stochastic variations in passenger demand (Daganzo and Ouyang, 2019). Efficient and robust metro operations require a modeling and optimization framework that has the capability of capturing the resulting complex system dynamics and deriving effective schedule control policies in real time.

* corresponding author.
E-mail address: andychow@cityu.edu.hk (A.H.F. Chow).

https://doi.org/10.1016/j.trb.2020.08.005
0191-2615/

There have been a number of studies presented in the literature investigating how one could improve the performances of train operations through service scheduling or rescheduling when incidents or perturbations occur. The integer programming or mixed integer programming formulations are among the most used approach in academia as they can be solved for global optimal solutions. For example, D'Ariano et al. (2007) present an alternative graph approach for rescheduling trains over a junction area and a solution algorithm based on the branch and bound method. Niu and Zhou (2013) formulate an integer programming for optimizing urban rail timetable under time-dependent demand. Barrena et al. (2014) present a set of exact integer programming formulations for scheduling train services with dynamic demand with solution algorithm based on the branch and cut method. Yang et al. (2016a) formulate a mixed integer programming model for scheduling high speed rail service with a solution algorithm implemented in CPLEX. Cacchiani et al. (2016) and Corman et al. (2017) adopt a mixed integer programming approach for real time train (re-)scheduling, and solve their optimization problems with self-designed heuristic methods. Despite their theoretical attractiveness, solving the integer or mixed integer programming models are usually computationally expensive even for medium-sized problems. Moreover, due to the complexity of the solution procedure, the integer or mixed integer programming approaches are mainly either for offline service scheduling, or for real time control for small scale service rescheduling purposes. Considering practical applications, there have also been a number of studies investigating the use of the computationally simpler heuristic approaches. This also includes use of some well-established meta-heuristics such as differential evolution (DE), genetic algorithm (GA, Niu and Zhou, 2013; Yang et al., 2017; Chow and Pavlides, 2018), and particle swarm optimization (PSO, Guo et al., 2017). A review and comparative study of these three meta-heuristics on train service scheduling can be found in Chang and Kwan (2004). Some other related studies can be found in Cacchiani et al. (2016), Corman et al. (2017), and Sama et al. (2017). More comprehensive review on different solution methods for train service scheduling can also be found in Yang et al. (2016b), Scheepmaker et al. (2017), and Zhang et al. (2018).

Train scheduling and rescheduling problems have been studied extensively over the past decade due to the increasing interest in green and sustainable transport modes around the globe. However, most related studies have been focusing on intercity and high speed rail trains instead of urban metro services. Moreover, most existing studies have been focusing on service scheduling for mid- or long-term average demand variation but do not respond to the short-term or real-time demand stochasticity. Existing work has not paid sufficient attention to scheduling algorithms that can be responsive to prevailing variations and stochasticity of passenger demand. With the advancements in information and communication technologies such as train positioning and smart card transaction systems, real time variations of service status and passenger demand can now be observed and reliably estimated (Shao et al., 2014; Zhou et al., 2017; Noursalehi et al., 2018). This offers opportunities and also calls for innovations on real-time demand-responsive metro service scheduling with advanced data analytics and optimization techniques (Sun et al., 2014; Robenek et al., 2016; Mo et al., 2019).

A demand-responsive train schedule can be derived from a Markov decision process (MDP, Bertsekas, 2019) with real time observations and estimates of passenger demand, service status, and operational constraints such as availability of rolling stock. This train service scheduling problem with respect to passenger demand variation is regarded as a dynamic optimization problem. It is known that the complexity of such dynamic optimization problem grows exponentially with the amount of decisions to make, state variables to observe, and uncertainties to consider. This issue is known as the curses of dimensionality which is a major technical challenge faced in most dynamic optimization problems (Powell, 2011; Bertsekas, 2019). There have been various techniques proposed to address the issue of curses of dimensionality. One of the prominent approaches is via the use of approximate dynamic programming (ADP) technique (Powell, 2011; Sutton and Barto, 2018). The core idea of ADP is to reduce the complexity of the original optimization problem by approximating the interaction between state-decision pairs and future costs with parametric approximators such as kernel regression models or artificial neural networks (ANNs). The optimization problem hence becomes working with parameters in the underlying approximator instead of dealing directly with the original state space. Examples of ADP application in railway operations include Yin et al. (2014) which adopt a rule-based expert system to develop an intelligent train operation algorithm that minimizes the energy consumption along the train's progression. Yin et al. (2016b) adopt an ADP to derive optimal train rescheduling strategies through parameterizing the state space with linear kernel regression functions. Liu et al. (2018) adopt a similar technique to derive an energy-efficient metro service schedule over a unidirectional service line with use of lookup tables as an approximation of the state-decision value function. Ghasempour and Heydecker (2020) apply the ADP idea to derive an optimal sequence of trains passing over a junction. Unfortunately, train scheduling is known to be non-convex and non-concave which implies it would not be straightforward to derive an optimal solution even with an approximated state-decision function. With approximating function for future costs with respect to prevailing state-decision pair, the previously mentioned studies solve for their optimal solution by exhaustive searches, which could take significant computational time and memory for problems involving vast decision space. Referring to the previous example we raised, Yin et al. (2014) address the issue by applying a local greedy search through a rule-based decision space for an optimal control policy. Yin et al. (2016b) address the issue by confining the search space through pre-identifying affected trains in their rescheduling problem. Liu et al. (2018) adopt a local greedy search for suboptimal solution. Ghasempour and Heydecker (2020) adopt a brute force search over one single junction. We would also like to point out that these previous studies only consider the expected or average performance delivered by their optimal schedules, but do not take into account of their robustness or performance variability under stochasticity. It should be emphasized that robustness of a control design is certainly an important aspect to investigate for applications in real time situation driven by stochasticity (Chow and Li, 2014; Li et al., 2019).

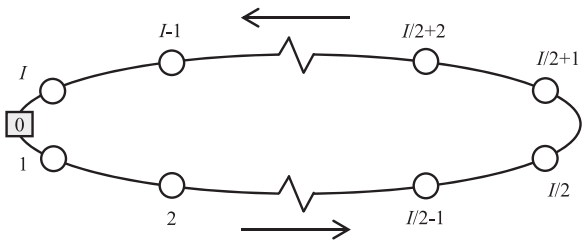


Fig. 1. Configuration of a metro service loop.

This study takes a significant step forward by proposing a real-time metro train scheduler for stochastic demand based upon an actor-critic deep reinforcement learning framework (Silver et al., 2016; Su et al., 2020; Lillicrap et al., 2015). It is noted that the proposed study focuses on the stochasticity on the demand side but not on the operational side such as capacity loss due to occurrence of incidents. Driven by a stochastic demand profile, the control agent schedules a set of service runs over a metro loop with circulation of limited rolling stock. The control agent considered herein is multi-objective which aims to minimize both passenger waiting cost and train operating cost over the service runs. At each decision stage, the control agent has to decide the dispatching time of the service, the dwell times at each station along the service route, and the running times over each station pair en-route. The total number of service runs to be dispatched is dependent on when all passengers can be served, which gives the scheduling task an infinite horizon optimal control problem. With the large state and decision spaces encountered, the state space is parameterized and represented by an artificial neural network (ANN) known as the 'critic' in the framework. The decision space is parameterized by another ANN known as the 'actor' in the framework. In general, the critic ANN estimates the future states and costs given the schedule decisions made, and in return the actor ANN delivers the corresponding schedule decisions with respect to the critic's estimates. The actor and critic ANNs will have to be trained via a series of simulated system transitions through a deep deterministic policy gradient (DDPG) learning algorithm before they can be applied for online schedule control. After sufficient training, our results show that the proposed actor-critic agent can outperform a selected set of meta-heuristics in terms of system efficiency and robustness over a range of stochastic demand scenarios. This study is among the first which integrates the emerging reinforcement learning technologies in computer science and dynamic transit system modeling in transportation.

The rest of paper is organized as follows. Section 2 presents the model formulation of the Markov schedule decision process. Section 3 presents the actor-critic deep reinforcement learning framework, including its architecture and training algorithm. In Section 4, the actor-critic schedule control agent is tested with a scenario configured with actual data collected from the Victoria Line of London Underground, UK. For benchmarking purpose, the control agent is compared against a selected set of meta-heuristics including differential evolution (DE), genetic algorithm (GA), and particle swarm optimization (PSO). The optimization algorithms are tested over a range of stochastic demand scenarios for both training and testing purposes. The results show that the actor-critic control agent can outperform the selected heuristics in terms of both system efficiency and robustness under stochastic demand-driven operational environments. Finally, Section 5 gives some concluding remarks and suggestions for future work.

2. Model formulation

This study models the metro train scheduling over a service loop as a Markov decision process (MDP). The service loop consists of a series of station nodes $\mathcal{I} = \{1, 2, \ldots, I\}$ (see Fig. 1). Each node i, where $i = 1, 2, \ldots, I$, is regarded as a platform for boarding and alighting passengers at a specific station. We further classify the set of station nodes $\mathcal{I}$ into $\mathcal{I}_{\mathcal{T}}$ and $\mathcal{I}_{\mathcal{M}}$, where $\mathcal{I}_{\mathcal{T}}$ is the subset of nodes in $\mathcal{I}$ that are designated as terminals, and $\mathcal{I}_{\mathcal{M}}$ are ordinary intermediate nodes. In addition to the station nodes $\mathcal{I}$, Node 0 in the figure represents the depot from where the trains are dispatched and to where the trains are returned. The representation herein is generic and is applicable to any settings with a depot dispatching and collecting returned trains. This includes a common case where one considers a bi-directional service on which nodes $i = 1, 2, \ldots, I/2$ could represent the sequence of stations along the outbound direction, and nodes $i = I/2 + 1, I/2 + 2, \ldots, I$ represents the sequence of stations along the inbound direction (see an example in Yin et al., 2016b). Each pair of nodes $(i, i + 1)$ is separated by a specific distance $d_{i,i+1}$. The optimization problem considers a fixed total fleet size of K trains to be assigned to N service loop runs over a specific period of time. If $K < N$, then some of the trains will have to be reassigned when they complete their previously assigned runs. The passenger demand for the metro service is represented by a series of time-dependent origin-destination matrices $\boldsymbol{O}_t = [O_{ij}(t)]$ which specify the demand rate (i.e. arriving passengers per unit time) in between each pair of station nodes i and j in $\mathcal{I}$ during time interval t over a loading period, $t \in [0, T]$. We consider this OD demand $\boldsymbol{O}_t$ to be random variables following a predefined (e.g. Gaussian, log-normal, Poisson) distribution for all times t in $[0, T]$. We would like to emphasize that the design of the MDP herein can work with any form of demand distribution specification.

In the following subsections, we start with presenting the stochastic transition process in the MDP, followed by the objective functions formulated, and the operational constraints imposed on the decision variables.

2.1. Stochastic transition

Given the infrastructure and demand settings, the train scheduling problem is regarded as a MDP in which each stage n refers to the service run n over the specified service loop with all station nodes $i \in \mathcal{I}$. Each stage n is characterized by the associated tuple of state variables $\boldsymbol{s}_n$, and influenced by the tuple of decision variables $\boldsymbol{x}_n$. The state tuple $\boldsymbol{s}_n = (\tau_{n,i}, \sigma_{n,i}, l_{n,i})$ at each stage n contains the arrival times $\tau_{n,i}$ of service run n, departure times $\sigma_{n,i}$ of service run n, and the number of passengers $l_{n,i}$ left by the departure of service run n over all stations $i \in \mathcal{I}$. It is noted each train can only carry up to a finite Λ passengers at all times. Passengers who cannot board service n due to overloading will have to stay at the station and wait for the next service $n+1$. The total number of residual passengers left by each service n at station i is recorded by the variable $l_{n,i}$ in $\boldsymbol{s}_n$.

Associated with the states, the decision variables $\boldsymbol{x}_n = (h_{n+1}, u_{n+1,i}, w_{n+1,i})$, for all $i \in \mathcal{I}$, are interpreted as the decisions made to service run $n+1$ based on observations in stage n (Powell, 2011). The variable h_{n+1} in $\boldsymbol{x}_n$ is the headway between the dispatching times of the current service run n and its follower $n+1$ from the depot; $w_{n+1,i}$ is the dwell times of service run $n+1$ at each station node i; $u_{n+1,i}$ is the running time of service $n+1$ over each station pair $(i, i+1)$. Different settings of decisions $\boldsymbol{x}_n$ influence the corresponding $\boldsymbol{s}_{n+1}$, and hence the overall performance of metro system in future stages. Considering the stochastic demand $\boldsymbol{O}_t$, the system dynamics herein is characterized by a stochastic transition function $P(\boldsymbol{s}_n, \boldsymbol{x}_n|\boldsymbol{O}_t)$ which proceeds from $\boldsymbol{s}_n$ to $\boldsymbol{s}_{n+1}$ with decisions $\boldsymbol{x}_n$ driven by a stochastic demand $\boldsymbol{O}_t$.

The terminal condition for the MDP is specified by a binary 0–1 indicator η_n. The variable η_n equals to 1 if all passengers in the system have been served by the end of stage n, and 0 if otherwise. Service runs (trains) will be kept being dispatched until η_N turns to 1 at some stage N, recognizing all passengers in the metro system have been served. Therefore, the number N is the total number of service runs needed to serve all passengers in the system and that will be dependent on the decision policy $\boldsymbol{x}_n$ and passenger demand $\boldsymbol{O}_t$.

We now present the state equations governing the evolution of the state variables given the decision variable settings. We first have the dispatching times $\sigma_{n,0}$ of all service runs n from the depot (node 0) based upon the dispatching headways h_{n+1} specified in $\boldsymbol{x}_n$ as (Newell and Potts, 1964; Chow et al., 2017; Chow and Pavlides, 2018; Daganzo and Ouyang, 2019):

$$\sigma_{n+1,0} = \sigma_{n,0} + h_{n+1}, \tag{1}$$

for all service runs (stages) $n = 1, 2, \ldots, N$. Moreover, with the departure time $\sigma_{n+1,i}$ of the train service $n+1$ from station i, where $i = 0, 1, 2, \ldots, I-1$, the corresponding arrival time $\tau_{n+1,i+1}$ of the service at the next station $i+1$ is derived with the assigned running time $u_{n+1,i}$ in $\boldsymbol{x}_n$ between station pairs $(i, i+1)$ as:

$$\tau_{n+1,i+1} = \sigma_{n+1,i} + u_{n+1,i}. \tag{2}$$

Following this, the departure time $\sigma_{n,i+1}$ of this train from the station $i+1$ is determined by adding up its assigned dwell time $w_{n,i+1}$ in $\boldsymbol{x}_n$ and other required durations, i.e.

$$\sigma_{n+1,i+1} = \tau_{n+1,i+1} + w_{n+1,i+1} + (\Delta_\tau)_{i+1}, \tag{3}$$

in which $(\Delta_\tau)_{i+1}$ is regarded as the required shunting time for trains arriving at station $i+1$. This can be for, say, turnaround or related maintenance operations should station $i+1$ be a terminal station (i.e. $i+1 \in \mathcal{I}_\mathcal{T}$) of the assigned service. This $(\Delta_\tau)_{i+1}$ can be zero if station $i+1$ is merely an intermediate station en-route (i.e. $i+1 \in \mathcal{I}_\mathcal{M}$).

The above summarizes the progression of trains along the service runs. Next we have to describe the evolution of the number of passengers on-board and at station. We first denote the total number of passengers *wanting* to board service $n+1$ at node i for node j along the service line as $\lambda_{n+1,i(j)}$. This $\lambda_{n+1,i(j)}$ can be derived as:

$$\lambda_{n+1,i(j)} = \int_{\sigma_{n,i}}^{\sigma_{n+1,i}} \hat{O}_{ij}(t)dt + l_{n,i(j)}, \tag{4}$$

which is dependent on the time lag (i.e., $\sigma_{n+1,i} - \sigma_{n,i}$) between the departure times of the preceding service n and the service $n+1$ (see also Chow et al., 2017; Chow and Pavlides, 2018). The notation $\hat{O}_{ij}(t)$ in the integrand above denotes a particular realization of $O_{ij}(t)$ based upon the specified distribution of $\boldsymbol{O}_t$. The second term $l_{n,i(j)}$ represents the number of residual passengers left at node i when the previous service n departed who would also like to travel to node j.

Given the passenger carrying capacity of each train Λ, the *actual* number of passengers $\hat{\lambda}_{n+1,i(j)}$ that could board service $n+1$ at node i for node j will be determined as:

$$\hat{\lambda}_{n+1,i(j)} = \min(\lambda_{n+1,i(j)}, b_{n+1,i(j)}\left[\Lambda - \rho_{n+1,i-1} + v_{n+1,i}\right]), \tag{5}$$

where

$$b_{n+1,i(j)} = \frac{\lambda_{n+1,i(j)}}{\sum_{m=i+1}^{I} \lambda_{n+1,i(m)}} \tag{6}$$

is the proportion of $\lambda_{n+1,i(j)}$ out of *all* passengers wanting to board the service $n+1$ at node i for all destinations. The notation $\nu_{n+1,i}$ in (5) refers to the number of passengers getting off from service $n+1$ at node i; $\rho_{n+1,i-1}$ is the number of passengers on-board when the service $n+1$ approaches station node i after leaving node $i-1$. The term $\left[\Lambda - \rho_{n+1,i-1} + \nu_{n+1,i}\right]$ hence gives the available space on service $n+1$ for accommodating boarding passengers at node i. The formulations presented in (5) and (6) allocate the available capacity Λ to passengers heading to different destinations according to the realized OD matrices $\hat{O}_{ij}(t)$. This allocation scheme is consistent with the multi-commodity FIFO (first-in-first-out) principle adopted in dynamic system modeling (see Tampére et al., 2011; Chow et al., 2017).

The occupancy $\rho_{n+1,i}$ when the train leaves station node i after loading and unloading the passengers can then be updated from the conservation principle through the following state equation for all intermediate stations $i \in \mathcal{I}_{\mathcal{M}}$:

$$\rho_{n+1,i} = \rho_{n+1,i-1} + \hat{\lambda}_{n+1,i} - \nu_{n+1,i}, \tag{7}$$

where $\hat{\lambda}_{n+1,i} = \sum_{m=i+1}^{I} \hat{\lambda}_{n+1,i(m)}$ is the *actual* total number of passengers for all nodes after i. It is noted that $\rho_{n+1,i} = 0$ for all terminals $i \in \mathcal{I}_{\mathcal{T}}$ as we assume that the trains are either initially empty at starting terminal, or all passengers will have to leave the train when the service arrives at the ending terminal.

We also have the state equation for the number of residual passengers left at node i following the departure of service run $n+1$, based upon the conservation principle as:

$$l_{n+1,i} = \sum_{m=i+1}^{I} (\lambda_{n+1,i(m)} - \hat{\lambda}_{n+1,i(m)}). \tag{8}$$

It is assumed the entire service loop is initially empty and hence we have the initial condition $l_{1,i(j)} = 0$ for the first service run $n = 1$ over all station nodes i and j.

Furthermore, we have

$$\nu_{n+1,j(i)} = \hat{\lambda}_{n+1,i(j)} \tag{9}$$

which is the portion of passengers alighting from service $n+1$ at node j who are from node i. Hence, the overall $\nu_{n,j}$ can be determined as:

$$\nu_{n+1,j} = \sum_{i=1}^{j-1} \nu_{n+1,j(i)} \tag{10}$$

The number of off-passengers determined from (10) can then be fed back into (5) and (7), and hence complete the entire evolution of number of passengers on-board and at platform over all intermediate station nodes.

Finally, we have the terminal condition indicator η_{n+1} for the MDP determined as

$$\eta_{n+1} = \begin{cases} 1, \text{ if } (\sigma_{n,I} \geq T) \cap (l_{n,i} = 0), \forall i = 1, 2, \ldots, I \\ 0, \text{ otherwise} \end{cases}. \tag{11}$$

As aforementioned, the indicator η_{n+1} is used to determine whether the MDP should be terminated for stage $n+1$ based upon the terminal conditions: whether the end of loading period T has been reached, and no residual passenger has been left at any station node by completion of service n.

The above completes the specifications of the entire transition from state $\boldsymbol{s}_n$ to state $\boldsymbol{s}_{n+1}$ given the associated train schedule and demand settings.

2.2. Objective functions

We now present the objective functions adopted in the optimizer for deriving the scheduling decisions $\boldsymbol{x}_n$. From the passengers' perspective, we first have a cost function representing the waiting cost incurred by passengers who are able to board the service n given a schedule decision setting $\boldsymbol{x}_n$. Fig. 2 illustrates the queueing process over arrivals and departures of train services at a specific station node (say, node i) in the form of cumulative arrivals and departures of passengers derived from a specific realized demand pattern $\hat{\boldsymbol{O}}_t$ from $\boldsymbol{O}_t$ (Daganzo and Ouyang, 2019). In the figure, the demand arriving process over time is recorded by the cumulative function $\Psi_i(t)$, which can be obtained from integrating the realized demand $\hat{O}_{ij}(t)$ originating from node i over time:

$$\Psi_i(t) = \int_0^t \sum_j \hat{O}_{ij}(t) dt. \tag{12}$$

The step function in the figure represents the cumulative departure over time (Daganzo and Ouyang, 2019). Each step in the departure step function represents the number of passengers that could be transported by the train (i.e. $\hat{\lambda}_{n,i}$) upon the departure of the train service (i.e. $\sigma_{n,i}$). Moreover, the vertical difference between the cumulative arrival and departure functions is the number of passengers left on the platform at the station (i.e. $l_{n,i}$) at the associated time.

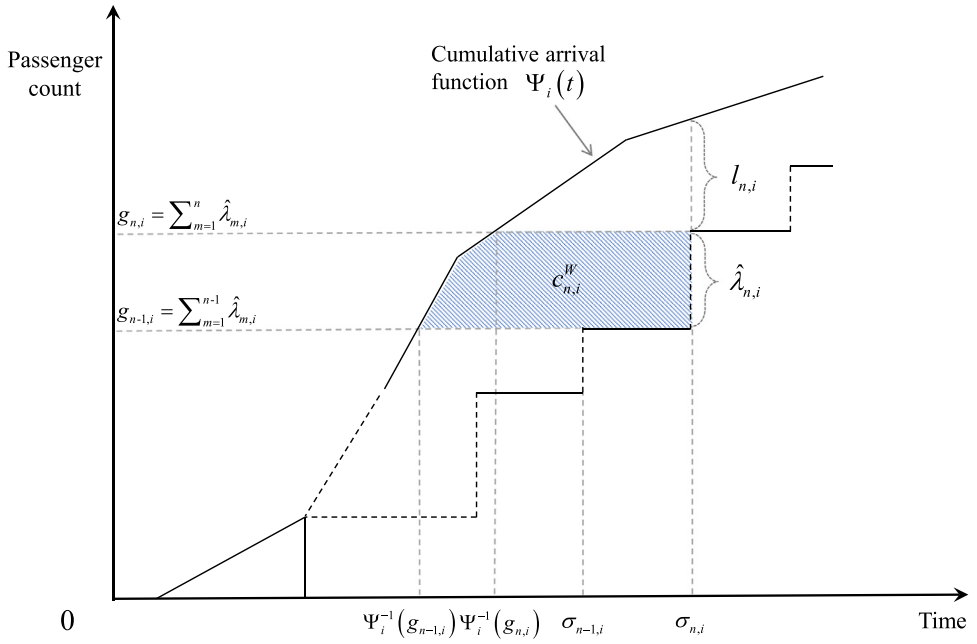


Fig. 2. Queueing process at a specific station node.

Let $g_{n,i}$ be the cumulative number of passengers who have been transported by the time of the departure of the service run n, which can be derived by summing up the number of passengers who boarded from service run $m = 1$ up to $m = n$:

$$g_{n,i} = \sum_{m=1}^{n} \hat{\lambda}_{m,i}. \tag{13}$$

Following classical queuing analysis (Daganzo and Ouyang, 2019; Chow and Pavlides, 2018), the total passengers' waiting time for service run n following the departure of run $n-1$ can be determined as the shaded area in Fig. 2 as:

$$c_{n,i}^{W} = \int_{g_{n-1,i}}^{g_{n,i}} \left[\sigma_{n,i} - \Psi_i^{-1}(g)\right] dg, \tag{14}$$

where g is an argument for cumulative passenger count, and $\Psi_i^{-1}(g)$ is an inverse function of $\Psi_i(t)$. We can also derive the monetary total waiting cost c_n^W of passengers over all station nodes i along the entire service route by train service n as (Chow and Pavlides, 2018):

$$c_n^W = \zeta^W \sum_{i=1}^{I} c_{n,i}^{W} \tag{15}$$

where the notation ζ^W in (15) represents an appropriate multiplier (e.g. value of time) which converts the total waiting time into monetary value according to DfT (2014) and Chow and Pavlides (2018). It is expected that the waiting cost is an increasing function with respect to the values of headways h_n and running times $u_{n,i}$ set in the schedule decisions $\boldsymbol{x}_n$.

From the operator's perspective, we consider the operating cost of the train services includes the energy-related and maintenance-related costs.

For the energy-related cost, we consider that the energy consumed by a train proceeding at a certain speed is for counteracting the associated resistance due to various factors including rolling, mechanical, and drag resistance (Li et al., 2013; Hansen et al., 2017). Given a train service n, the resistance force $F(v_{n,i})$ acted on the train that is running at a speed $v_{n+1,i} = \frac{d_{i,i+1}}{u_{n,i}}$ over track section between nodes i and $i+1$ can be estimated with the well-known Davis Equation (Davis, 1926; Hansen et al., 2017) as

$$F(v_{n,i}) = [\beta_1 + \beta_2 v_{n,i} + \beta_3 {v_{n,i}}^2](M_T + M_P \rho_{n,i}) \tag{16}$$

where M_T and M_P represent respectively the tare mass of the train and the average mass of a single passenger with personal belongings. The coefficients β_1, β_2, and β_3 are the model parameters for mass, mechanical, and air resistance which are dependent on the train characteristics. Assuming the tractive force applied to the train will be equal to the resistance force in (16) in order to maintain the train speed at $v_{n+1,i}$, we can derive the corresponding energy consumption for delivering this tractive force for train $n+1$ over the station node pair $(i, i+1)$ as the product of tractive force and the sectional

running distance. Incorporating the non-energy operating cost such as maintenance and depreciation, the operational cost c_n^V for train service (stage) n can be represented as (DfT, 2014):

$$c_n^V = \sum_{i=1}^{I} (\zeta^F F(v_{n,i}) d_{i,i+1} + \zeta^T u_{n,i} + \zeta^D d_{i,i+1}) \tag{17}$$

where the first term gives the energy cost as discussed above in which ζ^F is the unit cost of fuel; the second and third terms represent respectively the cost of speed and distance related operating expenditures, and ζ^T and ζ^D indicate respectively unit cost factor associated with running time and distance. The distance dependent cost in (17) can be regarded as a fixed cost for each train service. The operating cost (17) can be reduced by running slower trains (i.e. increasing $u_{n,i}$) and less trains (i.e. increasing $h_{n,i}$), which implies the two cost components considered herein ((15) and (17)) are conflicting objectives. It is noted that the focus of the current study lies in service scheduling and we do not consider high-order operational settings such as acceleration and braking profiles of trains. Consequently, the energy consumption term in (17) does not capture details associated with these operational settings. Incorporating these high-order operational settings requires more refined and well calibrated cost functions (see e.g. Li and Lo, 2014a; Li and Lo, 2014b; Yin et al., 2016a) and a more sophisticated underlying train simulation model that can capture the acceleration and braking process (see e.g. Umiliacchi et al., 2016). However, we would like to point out that the design of optimization framework herein is generic in terms of what objective functions, decision variables, and even the underlying train dynamics model are incorporated. We will leave the potential inclusion of objective function(s) for deriving detailed acceleration and braking profiles as a future study.

With the current two cost components, we can write down the total cost function for the schedule optimizer as:

$$c_n = \alpha c_n^W + (1-\alpha) c_n^V \tag{18}$$

where α is a coefficient adjusting the trade-off between passengers' cost and operator's cost. This gives the overall objective for the train scheduling considered in this study as:

$$\min_{\boldsymbol{x}_n} \mathbb{E}_{\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n | \boldsymbol{O}_t)} \left[\sum_{n=0}^{N} c_n \right], \tag{19}$$

through seeking an appropriate scheduling policy $\boldsymbol{x}_n^*$, for all $n = 0, 1, 2, \ldots, N$. The expectation in (19) is taken over the stochastic transition P subject to the demand distribution $\boldsymbol{O}_t$ as presented in Section 2.1.

2.3. Constraints

The schedule optimizer (19) is subject to a set of operational and regulatory constraints on $\boldsymbol{x}_n$. We first have upper and lower bounds imposed on the decision variables in $\boldsymbol{x}_n$:

$$h_{\min} \le h_n \le h_{\max}, \tag{20}$$

$$w_{\min} \le w_{n,i} \le w_{\max}, \tag{21}$$

$$(u_i)_{\min} \le u_{n,i} \le (u_i)_{\max}, \tag{22}$$

for all n and i, where $(u_i)_{\min}$ and $(u_i)_{\max}$ are the constrained bounds for running times in the section between the node pairs $(i, i+1)$ due to various engineering considerations including speed limit, train characteristics, and power supply. Similarly, $w_{\min}(w_{\max})$ and $h_{\min}(h_{\max})$ indicate respectively the lower(upper) bounds on the train dwell times and the service dispatching headways.

To maintain the level of service, the headway bounds are applicable for all service runs $n \ge 1$ over all station nodes i:

$$h_{\min} \le \sigma_{n+1,i} - \sigma_{n,i} \le h_{\max}. \tag{23}$$

It is noted that the dwell time of train at the terminals is taken as zero, i.e.

$$w_{n,i} = 0, \tag{24}$$

for all $n \ge 1$ and $i \in \mathcal{I}_{\mathcal{T}}$.

In addition, it is common in practical metro operations to prohibit a train from entering the platform before its predecessor has departed as a safety regulation (Chow and Pavlides, 2018). This safety constraint is implemented as:

$$\tau_{n+1,i} \ge \sigma_{n,i} + \Delta_S, \tag{25}$$

in which Δ_S is a predefined safety margin between any train service run pair $(n, n+1)$.

The optimizer also considers constraints due to the availability of rolling stock K, which is assumed to be less than the total number of service runs requested N. Following the dispatching order of the K trains to the service loop: for a service

run $n > K$, it is expected that the train that was used to serve run $n-K$ will have to be recirculated to this service run. This rolling stock circulation can be expressed in terms of the following inequality

$$\sigma_{n,0} \geq \tau_{n-K,I} + (\Delta_{\mathcal{T}})_I, \tag{26}$$

for all $n > K$. The left-hand-side in (26) is the scheduled dispatching time for service n, where $n > K$, and the right hand side is the time when the train for service n is available following the completion of its previously assigned service $n-K$ with inclusion of the shunting time involved. It is noted that constraint (26) is only applicable for $n > K$.

Finally, to resolve potential conflict between constraint (26) and the bounds (i.e. $h_{\max}$ and $h_{\min}$) imposed on the dispatching headways h_n for all n, we establish the following propositions. Proposition 2.1 derives the earliest dispatching times for service runs $n = 2, 3, \ldots, K$ following the first service run $n = 1$. Proposition 2.2 derives the feasible range for dispatching headways for service $n > K$.

Proposition 2.1. *The dispatching time $\sigma_{n,0}$ of service run $n = 2, 3, \ldots, K$ has to satisfy:*

$$\sigma_{n,0} \geq \tau_{1,I} + (\Delta_{\mathcal{T}})_I - (K+1-n)h_{\max}$$

Proof. Let's start with considering the train service run $n = K+1$. Following (1), the dispatching time of service run $n = K+1$ can be expressed as follows:

$$\sigma_{K+1,0} = \sigma_{K,0} + h_k.$$

Substituting above into (26), with $n = K+1$, gives:

$$\sigma_{K,0} + h_k \geq \tau_{1,I} + (\Delta_{\mathcal{T}})_I.$$

Given the bounds imposed on the dispatching headways as specified in (20), we can establish a lower bound on the dispatching time of service K using the inequality established above as:

$$\sigma_{K,0} \geq \tau_{1,I} + (\Delta_{\mathcal{T}})_I - h_{\max}.$$

Repeat the operation above by backward induction from $n = K$ toward $n = 2$ will lead to the following lower bound on the dispatching time of the service run $n = 2, 3, \ldots, K$ as:

$$\sigma_{n,0} \geq \tau_{1,I} + (\Delta_{\mathcal{T}})_I - (K+1-n)h_{\max}. \tag{27}$$

□

Proposition 2.2. *Suppose all operational constraints (i.e. (20)–(27)) established above are satisfied for a service run $n \geq K+1$. There exists a feasible headway h_{n+1} for dispatching the next service $n+1$ that satisfies both constraints (20) and (26). Moreover, the associated feasible range for this dispatching headway h_{n+1} will be $[\tau_{n-K+1,I} - \tau_{n-K,I}, h_{\max}]$.*

Proof. This is to identify the feasible range of h_{n+1} within $[h_{\min}, h_{\max}]$ as specified in (20) that makes the inequality (26) holds for service $n+1$ as

$$\sigma_{n+1,0} \geq \tau_{n-K+1,I} + (\Delta_{\mathcal{T}})_I.$$

The dispatching time for service $n+1$ can be expressed in terms of h_{n+1} by (1) as:

$$\sigma_{n+1,0} = \sigma_{n,0} + h_{n+1}.$$

If (26) holds for service run n, then we can have

$$\sigma_{n+1,0} \geq \tau_{n-K,I} + (\Delta_{\mathcal{T}})_I + h_{n+1}.$$

The expression above can be rewritten by adding and subtracting a term '$\tau_{n-K+1,I}$' on the right-hand-side as:

$$\sigma_{n+1,0} \geq (\tau_{n-K+1,I} + (\Delta_{\mathcal{T}})_I) + [h_{n+1} - (\tau_{n-K+1,I} - \tau_{n-K,I})].$$

To ensure (26) holds for service run $n+1$, we require the term $h_{n+1} - (\tau_{n-K+1,I} - \tau_{n-K,I})$ above to be non-negative, i.e.

$$h_{n+1} \geq \tau_{n-K+1,I} - \tau_{n-K,I}$$

where $\tau_{n-K,I}$ and $\tau_{n-K+1,I}$ are respectively returning times of consecutive services $n-K$ and $n-K+1$.

Combining this with the predefined upper bound $h_{\max}$ on dispatching headways, we have

$$\tau_{n-K+1,I} - \tau_{n-K,I} \leq h_{n+1} \leq h_{\max}, \tag{28}$$

and this gives the feasible range on h_{n+1} for constraints (20) and (26) to hold. □

3. Deep reinforcement learning based on actor-critic framework

This section introduces the online implementation of the Markov scheduling decision process developed in the previous section. We first define Ω_n to be the stage-dependent feasible set of $\boldsymbol{x}_n$ at stage n. We regard a schedule decision $\boldsymbol{x}_n$ in stage n is feasible, i.e. $\boldsymbol{x}_n \in \Omega_n$, if $\boldsymbol{x}_n$ satisfies (20)–(28), and stage n has not reached the terminal criterion (i.e. $\eta_n = 0$; see (11)). As aforementioned, the decision process will be terminated at some stage N when $\eta_N = 1$, indicating that all passengers loaded into the system would have been served at that stage N.

Based on the modeling framework we develop, the online schedule optimizer aims to seek a $\boldsymbol{x}_n^* \in \Omega_n$ at each stage n in order to minimize the total cost from that stage onward to a stage $n = N$ when all passengers could be served in metro system, i.e.

$$\boldsymbol{x}_n^* = \arg\min_{\boldsymbol{x}_n \in \Omega_n} \mathbb{E}_{\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n | \boldsymbol{O}_t)} \left[\sum_{m=n}^{N} \gamma^{m-n} (1 - \eta_n) c_n \right]. \tag{29}$$

The online minimization objective is subject to the stochastic transition $\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n | \boldsymbol{O}_t)$ established in the decision process. A discount factor $\gamma \in (0, 1]$ is associated with each future stage m, where $m \geq n$, which determines how the cost at each of these future stages m is valued (Powell, 2011; Bertsekas, 2019). The stage costs are summed up to N in (29) where $\eta_N = 1$.

Expression (29) can be rewritten as

$$\boldsymbol{x}_n^* = \arg\min_{\boldsymbol{x}_n \in \Omega_n} \mathbb{E}_{\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n | \boldsymbol{O}_t)} [c_n + \gamma (1 - \eta_n) V(\boldsymbol{s}_{n+1})], \tag{30}$$

in which $V(\boldsymbol{s}_{n+1})$ is known as the value function which represents the total optimal cost from stage $n+1$ onward to stage N given the state at stage $n+1$ is $\boldsymbol{s}_{n+1}$. The value function can be written recursively following the well-established Bellman equation (Powell, 2011; Bertsekas, 2019) as:

$$V(\boldsymbol{s}_n) = \min_{\boldsymbol{x}_n \in \Omega_n} \mathbb{E}_{\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n | \boldsymbol{O}_t)} [c_n + \gamma (1 - \eta_n) V(\boldsymbol{s}_{n+1})]. \tag{31}$$

Solving (30) for $\boldsymbol{x}_n^*$ faces two major computational challenges. We first need to obtain $V(\boldsymbol{s}_{n+1})$ which requires visiting all state tuples $\boldsymbol{s}_n = (\tau_{n,i}, \sigma_{n,i}, l_{n,i})$ over all stations $i = 1, 2, \ldots, I$, over all future stages $n+1, n+2, \ldots, N$, over all potential demand realizations generated by $\boldsymbol{O}_t$, and evaluating all combinations with all admissible decisions $\boldsymbol{x}_n \in \Omega_n$. Hence, obtaining the exact $V(\boldsymbol{s}_{n+1})$ can be extremely computationally demanding, if possible, even for medium-sized problems. Even if one manages to obtain $V(\boldsymbol{s}_{n+1})$, the second challenge lies in the determination of the corresponding solution $\boldsymbol{x}_n^* = (h_{n+1}^*, u_{n+1,i}^*, w_{n+1,i}^*)$ from the infinitely many possible train run trajectories subject to the generic demand distribution $\boldsymbol{O}_t$. We should also note that N herein is dependent on when all passenger can be eventually served (i.e. when $\eta_n = 1$). Hence, N cannot be known in advance and it is dependent on the evolution of state-decision pair driven by the stochastic demand $\boldsymbol{O}_t$. Together with the uncertainties in $\boldsymbol{O}_t$, the computational issue identified herein is known as the curses of dimensionality in the literature of dynamic programming (Powell, 2011; Bertsekas, 2019), and is most often-cited reason why service scheduling problems in a stochastic environment can be computationally intractable and not able to be solved readily.

Considering the issue of the curses of dimensionality, this study proposes an actor-critic deep reinforcement learning framework for solving the scheduling decision problem. An optimal schedule $\boldsymbol{x}_n^*$ is to be derived by a control agent based upon prevailing states $\boldsymbol{s}_n$. The following subsections will introduce the incorporation of constraints in the reinforcement learning framework, the proposed actor-critic architecture, and the training algorithm for the actor-critic schedule control agent.

3.1. Incorporation of constraints

The basic reinforcement learning framework does not consider explicitly the constraints imposed on the decision space (Li, 2017). To incorporate the constraints set up in Section 2.3, we formulate an embedding decision tuple $\bar{\boldsymbol{x}}_n = (\bar{h}_{n+1}, \bar{u}_{n+1,i}, \bar{w}_{n+1,i})$ for all stages n which is to be used in the actor-critic reinforcement learning framework. The embedding decision tuple has exactly the same dimension as the actual decision tuple $\boldsymbol{x}_n = (h_{n+1}, u_{n+1,i}, w_{n+1,i})$, except that all elements in $\bar{\boldsymbol{x}}_n$ are defined in the interval $[-1, 1]$. The elements in $\bar{\boldsymbol{x}}_n$ can be decoded back into their counterparts in $\boldsymbol{x}_n$ through the following procedure.

We first define a linear mapping f for incorporating the fixed bounds specified in each of the constraints (20)–(22), i.e.

$$h_n := f(\bar{h}_n) = \frac{2(\bar{h}_n - h_{\min})}{h_{\max} - h_{\min}} - 1 \tag{32}$$

$$w_{n,i} := f(\bar{w}_{n,i}) = \frac{2(\bar{w}_{n,i} - w_{\min})}{w_{\max} - w_{\min}} - 1 \tag{33}$$

$$u_{n,i} := f(\bar{u}_{n,i}) = \frac{2[\bar{u}_{n,i} - (u_i)_{\min}]}{(u_i)_{\max} - (u_i)_{\min}} - 1 \tag{34}$$

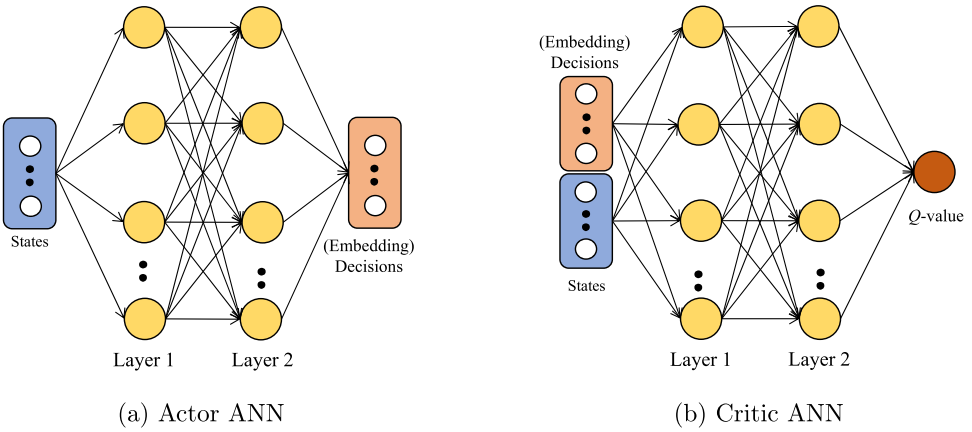


Fig. 3. Structure of the artificial neural networks (ANNs) adopted in the actor-critic deep reinforcement learning framework.

To incorporate the stage-dependent constraints specified through (23)–(28), we adopt the following operator on each element x contained in $\boldsymbol{x}_n$:

$$x := \text{clip}(f(\bar{x}), (x_{(n)})_{\min}, (x_{(n)})_{\max}), \tag{35}$$

where $\bar{x}$ is the corresponding element in $\tilde{\boldsymbol{x}}_n$ representing x; $(x_{(n)})_{\min}$ and $(x_{(n)})_{\max}$ are respectively the stage-dependent lower and upper bounds on x in $\boldsymbol{x}_n$ improved through (23)–(28) during stage n. The stage-dependent bounds $((x_{(n)})_{\min}, (x_{(n)})_{\max})$ herein can be obtained for each x through running the system transition $\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n | \boldsymbol{O}_t)$ over each n as driven by the sampled demand settings $\hat{O}_{ij}(t)$.

The constructed embedding decision tuple $\tilde{\boldsymbol{x}}_n$ set up herein will be used in place of $\boldsymbol{x}_n$ in the deep reinforcement learning framework which is to be presented in the following section.

3.2. Actor-critic architecture

The major challenge in solving (30) is the computation of the value function V_{n+1} and the corresponding $\boldsymbol{x}_n^*$ via minimizing the expectation on the right-hand-side of (30). To address the computational issue, we set up two two-layer artificial neural networks (ANNs, see Fig. 3) which represent respectively the 'actor' and the 'critic' in the actor-critic framework. The actor and critic ANNs are used respectively for approximating the value function and generating the decisions $\boldsymbol{x}_n^*$ based upon prevailing states. Both actor and critic ANNs consist of multiple linear layers and element-wise non-linear activation functions (e.g. rectified linear unit (ReLU) and hyperbolic tangent (Tanh); see Goodfellow et al., 2016). Mathematically, the ANN can be regarded as a generalized form of the linear kernel regression model adopted in many related work with use of approximate dynamic programming (see Yin et al., 2016b; Liu et al., 2018, and Ghasempour and Heydecker, 2020). In particular, the ANN covers the linear kernel regression model as a special (and simpler) case with single layer of few elements (known as 'neurons' in ANN terminology, see Bertsekas, 2019). With its sophisticated mathematical structure, the advantage of ANN over the linear kernel regression model is that it can be more sophisticated system dynamics, in particular those driven by the surrounding stochasticity such as the stochastic demand flows considered herein. The significance of the capability of capturing the dynamic and stochastic characteristics in the proposed deep reinforcement learning framework is to be demonstrated through the numerical experiments in the following section.

The actor ANN is denoted by $\pi_{\boldsymbol{\phi}}$ and it is specified by parameter set $\boldsymbol{\phi}$. Given the prevailing state $\boldsymbol{s}_n$ at stage n, the actor ANN generates a corresponding (embedding) decision $\tilde{\boldsymbol{x}}_n$ which aims to minimize the total cost from stage n onward as specified in (30).

The critic ANN (see Fig. 3b) is denoted by $Q_{\boldsymbol{\theta}}$ and it is specified by parameter set $\boldsymbol{\theta}$. The use of the critic ANN is analogous to the Q-learning in the literature of approximate dynamic programming (Powell, 2011; Yin et al., 2016b). Given the prevailing state $\boldsymbol{s}_n$ and the (embedding) decision $\tilde{\boldsymbol{x}}_n$ derived by the actor ANN, the critic ANN generates an estimate $Q_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \tilde{\boldsymbol{x}}_n)$ for the Q-value $Q(\boldsymbol{s}_n, \tilde{\boldsymbol{x}}_n)$. The Q-value is defined as the total expected future cost from stage n onward given the prevailing states and (embedding) decisions are $\boldsymbol{s}_n$ and $\tilde{\boldsymbol{x}}_n$ respectively.

By definition, we have (Powell, 2011)

$$V(\boldsymbol{s}_n) = \min_{\tilde{\boldsymbol{x}}_n} Q(\boldsymbol{s}_n, \tilde{\boldsymbol{x}}_n). \tag{36}$$

Following the Bellman Eq. (31), $Q(\boldsymbol{s}_n, \tilde{\boldsymbol{x}}_n)$ can be expressed recursively upon decision $\tilde{\boldsymbol{x}}_n$ as (Lillicrap et al., 2015):

$$Q(\boldsymbol{s}_n, \tilde{\boldsymbol{x}}_n) = \mathbb{E}_{\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \tilde{\boldsymbol{x}}_n | \boldsymbol{O}_t)}[c_n + \gamma(1-\eta_n)\mathbb{E}_{\tilde{\boldsymbol{x}}_{n+1}} Q(\boldsymbol{s}_{n+1}, \tilde{\boldsymbol{x}}_{n+1})]. \tag{37}$$

Should the schedule decision policy $\bar{\boldsymbol{x}}$ be deterministic throughout the transition process, the inner expectation on $\bar{\boldsymbol{x}}_{n+1}$ in (37) can then be avoided and the state Eq. (37) can be reduced to (Lillicrap et al., 2015):

$$Q(\boldsymbol{s}_n, \bar{\boldsymbol{x}}_n) = \mathbb{E}_{\boldsymbol{s}_{n+1}\sim P(\boldsymbol{s}_n,\bar{\boldsymbol{x}}_n|\boldsymbol{O}_t)}[c_n + \gamma(1-\eta_n)Q(\boldsymbol{s}_{n+1}, \bar{\boldsymbol{x}}_{n+1})]. \tag{38}$$

The parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ will first have to be determined before the ANNs can be used for delivering effective train schedules in online applications. The state Eq. (38) plays an important role in the training algorithm for this actor-critic framework which is to be presented in the following subsection.

3.3. Training algorithm via deep deterministic policy gradient

The ANNs will be trained by the deep deterministic policy gradient (DDPG) algorithm (Lillicrap et al., 2015). The DDPG algorithm is a model-free training approach (Powell, 2011) which trains the actor-critic agent from realized transitions without needing any prior knowledge of the underlying transition model. Moreover, DDPG is an off-policy learning method that considers the schedule decision policy $\bar{\boldsymbol{x}}$ to be deterministic, and is determined from the prevailing states $\boldsymbol{s}$ through a specific mapping π.

The procedure of the DDPG training process is summarized in Algorithm 1. The training process starts with setting

Algorithm 1 DDPG training algorithm for the actor and critic ANNs in the reinforcement learning framework.

Require: Initialize the actor ANN $\pi_{\boldsymbol{\phi}}$ and the critic ANN $Q_{\boldsymbol{\theta}}$
Ensure: Trained actor ANN $\pi_{\boldsymbol{\phi}}$ for generating schedule decision that minimizes (19)

Initialize the target ANN parameters: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$, $\boldsymbol{\phi}' \leftarrow \boldsymbol{\phi}$
Initialize the replay buffer with Z transitions generated by a random policy
for $r \leftarrow 1$ to R **do**
  Initialize stage index: $n \leftarrow 0$
  Initialize the MDP with $\boldsymbol{s}_0 \leftarrow \boldsymbol{0}$, $\eta_0 \leftarrow 0$, and sample a $\hat{\boldsymbol{O}}_t$ from its predefined distribution
  while $\eta_n = 0$ **do**
    Get the embedding decision : $\bar{\boldsymbol{x}}_n = \text{clip}(\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_n) + \boldsymbol{\chi}, -1, 1)$, where $\boldsymbol{\chi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\epsilon}_r)$
    Decode the actual decision: $\boldsymbol{x}_n = \text{clip}(f(\bar{\boldsymbol{x}}_n), (\boldsymbol{x}_n)_{\min}, (\boldsymbol{x}_n)_{\max})$
    Execute $\boldsymbol{x}_n$ for the corresponding c_n, $\boldsymbol{s}_{n+1}$, and η_n via transition $P(\boldsymbol{s}_n, \boldsymbol{x}_n|\boldsymbol{O}_t)$
    Insert the newly acquired transition experience $(\boldsymbol{s}_n, \bar{\boldsymbol{x}}_n, c_n, \boldsymbol{s}_{n+1}, \eta_{n+1})$ into the replay buffer
    Sample a batch of B transitions $(\boldsymbol{s}_b, \bar{\boldsymbol{x}}_b, c_b, \boldsymbol{s}_{b+1}, \eta_{b+1})$, $b = 1, 2, \ldots, B$, from the replay buffer
    Calculate the target Q-values $y_b = c_b + \gamma(1-\eta_b)Q_{\boldsymbol{\theta}'}(\boldsymbol{s}_{b+1}, \pi_{\boldsymbol{\phi}'}(\boldsymbol{s}_{b+1}))$, for all $b = 1, 2, \ldots, B$
    Determine new $\boldsymbol{\theta}$ in the critic ANN as:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_Q(\boldsymbol{\theta}) = \frac{1}{B}\sum_b (Q_{\boldsymbol{\theta}}(\boldsymbol{s}_b, \bar{\boldsymbol{x}}_b) - y_b)^2$$

    Determine new $\boldsymbol{\phi}$ in the actor ANN as

$$\boldsymbol{\phi} = \arg\min_{\boldsymbol{\phi}} \mathcal{L}_A(\boldsymbol{\phi}) = \frac{1}{B}\sum_b Q_{\boldsymbol{\theta}^*}(\boldsymbol{s}_b, \pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b))$$

  via the search gradient:

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_A(\boldsymbol{\phi}) = \mathbb{E}[\nabla_{\bar{\boldsymbol{x}}_b} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_b, \bar{\boldsymbol{x}}_b)|_{\bar{\boldsymbol{x}}_b=\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b)} \nabla_{\boldsymbol{\phi}}\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b)]$$

    Update the target parameters of the ANNs:

$$\boldsymbol{\theta}' \leftarrow \xi\boldsymbol{\theta} + (1-\xi)\boldsymbol{\theta}'$$

$$\boldsymbol{\phi}' \leftarrow \xi\boldsymbol{\phi} + (1-\xi)\boldsymbol{\phi}'$$

    Update the stage index: $n \leftarrow n+1$
  end while
  Update the explorative noise $\boldsymbol{\epsilon}^r = \boldsymbol{\epsilon}^{\min} + \frac{1}{2}(\boldsymbol{\epsilon}^{\max} - \boldsymbol{\epsilon}^{\min})(1 + \cos(\min(\frac{r}{R_{\boldsymbol{\epsilon}}}, 1)\pi))$
end for

up a replay buffer which is a data queue of size Z for storing realized transition samples $(\boldsymbol{s}_n, \bar{\boldsymbol{x}}_n, c_n, \boldsymbol{s}_{n+1}, \eta_{n+1})$ between successive stage pairs in a 'first-in-first-out' (FIFO) manner over a series of simulation runs. The replay buffer serves as a dynamic library of ground truths for training the ANNs over epoch $r = 1, 2, \ldots, R$.

It is noted that deep reinforcement learning process can be unstable when a highly nonlinear approximator (e.g. the artificial neural network) is used to represent the Q- and policy functions (Mnih et al., 2015). An update to the parameters $\boldsymbol{\theta}$ in the critic ANN $Q_{\boldsymbol{\theta}}$ may affect the correlation between its estimated Q-value $Q_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \bar{\boldsymbol{x}}_n)$ and the associated target value $[c_n + \gamma(1-\eta_n)Q_{\boldsymbol{\theta}}(\boldsymbol{s}_{n+1}, \bar{\boldsymbol{x}}_{n+1})]$ established in the Bellman Eq. (38). To ensure stability in the training process, we define $\boldsymbol{\phi}'$

and $\boldsymbol{\theta}'$ as the target parameters (Lillicrap et al., 2015; Mnih et al., 2015) in the actor ANN and critic ANN respectively. The target parameters will be initialized as: $\boldsymbol{\phi}' \leftarrow \boldsymbol{\phi}$ and $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$.

In each training epoch r, a specific OD-demand $\hat{\boldsymbol{O}}_t$ will first be sampled from the assumed distribution of $\boldsymbol{O}_t$. An embedding decision $\bar{\boldsymbol{x}}_n$ will then be generated by the actor ANN $\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_n)$ based upon the prevailing state $\boldsymbol{s}_n$ and parameters $\boldsymbol{\phi}$. To avoid being attracted to an inefficient decision policy, we formulate a Gaussian noise tuple $\boldsymbol{\chi}$ which has the same dimension as $\bar{\boldsymbol{x}}_n$ (and hence $\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_n)$). Each element in $\boldsymbol{\chi}$ follows a Gaussian distribution with zero mean and a finite standard deviation ϵ_r, where

$$\epsilon^r = \epsilon^{\min} + \frac{1}{2}(\epsilon^{\max} - \epsilon^{\min})(1 + \cos(\min(\frac{r}{R_\epsilon}, 1)\pi)), \tag{39}$$

in which $\epsilon^{\min}$ and $\epsilon^{\max}$ are respectively the lower and upper bounds set on ϵ_r; R_ϵ is the epoch at which $\boldsymbol{\chi}$ would stop further decaying. This ϵ^r is known as the explorative noise and its standard deviation is set to be decreasing over training epoch r for convergence (Loshchilov and Hutter, 2016).

The Gaussian $\boldsymbol{\chi}$ noise tuple is added to the embedding decision tuple $\bar{\boldsymbol{x}}_n$ as

$$\bar{\boldsymbol{x}}_n = \text{clip}(\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_n) + \boldsymbol{\chi}, -1, 1). \tag{40}$$

The introduction of this ϵ_r is to encourage more random exploration (with larger ϵ_r) in early stages of the training process in order to avoid over-fitting to a specific realized demand setting, but more refined exploitation (with smaller ϵ_r) in later ones in order to ensure stability and convergence (Powell, 2011). The actual decisions $\boldsymbol{x}_n$ will be decoded from the embedding decisions $\bar{\boldsymbol{x}}_n$ through (35), and be executed through the transition $P(\boldsymbol{s}_n, \boldsymbol{x}_n|\boldsymbol{O}_t)$ with realized OD demand $\hat{\boldsymbol{O}}_t$ for the corresponding cost c_n, state $\boldsymbol{s}_{n+1}$, and terminal indicator η_{n+1}. The newly acquired transition experience $(\boldsymbol{s}_n, \bar{\boldsymbol{x}}_n, c_n, \boldsymbol{s}_{n+1}, \eta_{n+1})$ will be pushed into and update the replay buffer queue.

A batch of B transitions will then be sampled from the updated replay buffer, with which we will calculate the target value y_b of the Q-value associated with each of the sampled transitions b as:

$$y_b = c_b + \gamma(1 - \eta_b)Q_{\boldsymbol{\theta}'}(\boldsymbol{s}_{b+1}, \pi_{\boldsymbol{\phi}'}(\boldsymbol{s}_{b+1})), \tag{41}$$

for all $b = 1, 2, \ldots, B$ in the sampled batch.

The critic ANN $Q_{\boldsymbol{\theta}}$ is to be updated with respect to these target values derived from the sampled batch. The concept herein aligns with the one-step temporal difference (TD) learning in the reinforcement learning literature (Sutton and Barto, 2018). The critic ANN update is carried out via the following stochastic batch optimization (Goodfellow et al., 2016) which seeks a new $\boldsymbol{\theta}$ that minimizes the mean squared error $\mathcal{L}_Q$ between each of the targets y_b and the corresponding Q-values in the batch:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_Q(\boldsymbol{\theta}) = \frac{1}{B}\sum_{b=1}^{B}(Q_{\boldsymbol{\theta}}(\boldsymbol{s}_b, \bar{\boldsymbol{x}}_b) - y_b)^2. \tag{42}$$

With the new $\boldsymbol{\theta}$ acquired, the actor ANN is also to be updated by seeking a new $\boldsymbol{\phi}$ that minimizes the expected Q-value generated by the updated critic ANN over all state-decision pairs $(\boldsymbol{s}_b, \boldsymbol{x}_b)$ in the sampled batch as:

$$\boldsymbol{\phi} = \arg\min_{\boldsymbol{\phi}} \mathcal{L}_A(\boldsymbol{\phi}) = \frac{1}{B}\sum_{b} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_b, \pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b)). \tag{43}$$

The minimization (43) can be solved by a stochastic gradient search with the gradient $\nabla_{\boldsymbol{\phi}}\mathcal{L}_A(\boldsymbol{\phi})$ conditioned on $\boldsymbol{\phi}$. The gradient can be derived following the chain rule and the deterministic policy gradient theorem proposed by Silver et al. (2014) as

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_A(\boldsymbol{\phi}) = \mathbb{E}[\nabla_{\bar{\boldsymbol{x}}_b} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_b, \bar{\boldsymbol{x}}_b)|_{\bar{\boldsymbol{x}}_b = \pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b)} \nabla_{\boldsymbol{\phi}}\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b)] \tag{44}$$

in which the terms on right hand side takes expectation over the multiplication of the gradient of critic ANN on the decision embedding (which equals to $\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_b)$) and the gradient of actor ANN $\pi_{\boldsymbol{\phi}}$. There are effective algorithms available for solving the optimization problems defined in (42) and (43), such as the well established Adam algorithm (Kingma and Ba, 2014; Alom et al., 2019) which is to be used for the present study.

To ensure the convergence and stability of the training process (Mnih et al., 2015), the target parameters in the ANNs are to be updated via the moving averaging:

$$\boldsymbol{\theta}' \leftarrow \xi\boldsymbol{\theta} + (1 - \xi)\boldsymbol{\theta}' \tag{45}$$

$$\boldsymbol{\phi}' \leftarrow \xi\boldsymbol{\phi} + (1 - \xi)\boldsymbol{\phi}' \tag{46}$$

where ξ is a sufficiently small smoothing factor.

The training process will be repeated for R times. With the decaying explorative noise ϵ^r over training epochs r, the replay buffer will gradually be filled up with transition experiences with decisions $\bar{\boldsymbol{x}}_n$ converging toward $\pi_{\boldsymbol{\phi}}(\boldsymbol{s}_n)$.

Table 1
Configuration of the case study service line.

Track Section	Distance (m)	Minimum running time (s)	
		Outbound	Inbound
Brixton (BXN) — Stockwell (SKW)	1460	130	121
Stockwell (SKW) — Vauxhall (VXL)	1770	133	138
Vauxhall (VXL) — Pimlico (PCO)	800	82	84
Pimlico (PCO) — Victoria (VIC)	1190	130	109
Victoria (VIC) — Green Park (GPK)	1090	117	112
Green Park (GPK) — Oxford Circus (OXC)	1140	118	106
Oxford Circus (OXC) — Warren Street (WRR)	900	91	103

4. Numerical experiments

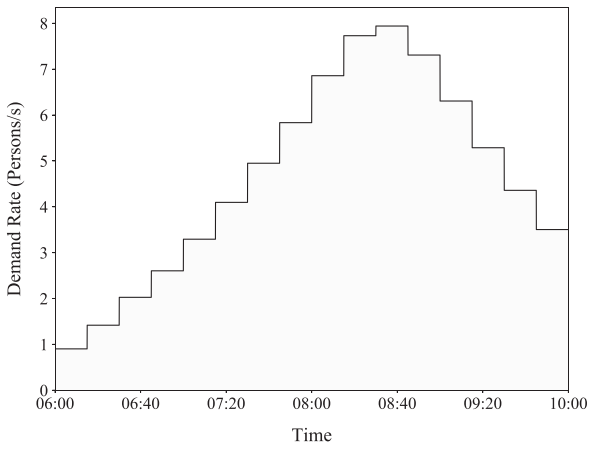
4.1. Experiment settings

The actor-critic reinforcement learning framework is now tested with a real-world scenario configured with data collected from the Victoria Line of London Underground, UK. We consider a four-hour service duration from 06:00 to 10:00 on a typical weekday. Trains are to be dispatched from a depot to a service loop consisting of eight stations with the two terminals at Brixton (BXN) and Warren Street (WRR). This gives a total of $I = 16$ station nodes along the service line including both outbound and inbound directions. The depot is located at the Brixton station and a total of $K = 24$ trains with a carrying capacity of $\Lambda = 500$ passengers are available for dispatching. At each terminal (i.e. $i = 8$; $i = 16$), it takes a shunting duration of $\Delta_{\mathcal{T}} = 218s$ to complete the dwelling, maintenance, and turnaround operations. The distances $d_{i,i+1}$ and minimum running times $(u_i)_{\min}$ between each station pair $(i, i+1)$ are given in Table 1. The maximum running times $(u_i)_{\max}$ are determined by adding a 30s buffer to the minimum running times $(u_i)_{\min}$ for energy saving and unexpected disturbances. Considering practical implementation, the dispatching headways and service dwell times are bounded in the intervals [100, 500]s and [25, 50]s respectively. The safety headway margin between each successive pair of train runs is set as $\Delta_S = 60s$.

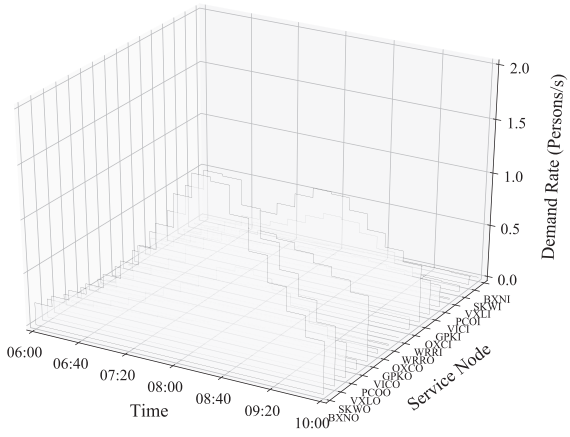
The OD demand $\boldsymbol{O}_t$ adopted herein is inferred from the rolling origin and destination survey (RODS) conducted by the Transport for London. Fig. 4 shows the overall average passenger arrival rates (persons per second) over the entire service line (4 a), and at each station nodes along both inbound (stations associated with suffix 'I') and outbound (stations associated with suffix 'O') directions (4 b). To incorporate the stochasticity, the OD demand $\boldsymbol{O}_t$ is assumed to follow a multivariate Gaussian distribution over each time interval t with a mean value equaling to the average values recorded in the RODS dataset. The use of the Gaussian distribution model is due to its popularity in travel demand analysis (see Shao et al., 2014; Yang et al., 2018; Noursalehi et al., 2018), and its flexibility of capturing spatial-temporal correlation with appropriate settings of the underlying covariance matrix. Nevertheless, we would like to reemphasize the proposed optimization framework can work with any form of demand distribution model. To investigate the effect of stochasticity, we consider three demand scenarios with different levels of uncertainty. The first is a deterministic demand profile with zero stochastic variation (D00). The second scenario considers each demand rate in $\boldsymbol{O}_t$ is associated with a standard deviation equaling to 25% of its mean value (R25). The third scenario is each demand rate in $\boldsymbol{O}_t$ is associated with a standard deviation equaling to 50% of the mean value (R50).

For the objective functions, we first set the trade-off coefficient α in (18) to be 0.5. Effect of choosing different values for α will be presented in Section 4.6. The passenger and operational costs are to be valued following the guidance set by the UK Department for Transport (DfT, 2014). The waiting time of passengers is valued as $\zeta^W = 4.486 \times 10^{-3} £/s$. The distance- and time-based unit operational cost are set as $\zeta^D = 3.046 \times 10^{-4} £/m$ and $\zeta^T = 1.929 \times 10^{-3} £/s$ respectively. As aforementioned, the energy-related cost is to be estimated by the Davis formula, assuming the energy consumed by a train is for counteracting the associated resistance (Li et al., 2013). To specify the Davis formula, we take the tare mass of a service train as $M_T = 197.30$ metric tons and the average weight of a passenger (with their personal items) as $M_P = 0.08$ metric tons (=80 kg). The unit cost associated with the energy (electricity) consumption is taken as $\zeta^F = 0.1077 £/kWh$, and the Davis function coefficients are set to be $\beta_1 = 16.600 kN/\text{ton}$, $\beta_2 = 0.366 kN/\text{ton}(m/s)$, and $\beta_3 = 0.026 kN/\text{ton}(m/s)^2$.

For the architecture of the actor-critic deep reinforcement learning framework, both actor and critic ANNs have two layers consisting of 400 and 300 neurons respectively (see Fig. 3), with a ReLU activation unit between each layer for both actor and critic ANNs. The output layer of the actor ANN (π_ϕ) is followed by a Tanh activation unit such that the generated embedding decision outputs $\tilde{\boldsymbol{x}}_n$ can be constrained over the continuous interval [-1, 1]. The schedule control agent is to be trained by the DDPG algorithm before being applied online. To facilitate the training process, the parameters in the ANNs are to be initialized by the Xavier initialization method (Glorot and Bengio, 2010). An experience replay buffer of size $Z = 50,000$ is initialized with transition experiences generated by an arbitrary scheduling policy. The batch size of the sampled transitions from the replay buffer for updating the ANN parameters is set to be $B = 100$. The discount factor γ for computing the Q-values of the sampled transitions (y_b) is set to be 0.99. The smoothing factor ξ for updating the ANN parameters (θ and ϕ) at the end of each training epoch is set to be 5×10^{-3}. The total number of training epochs is set to



(a) Average demand rate over the service line



(b) Average demand rate at each station node

Fig. 4. Passenger demand profile along the Victoria Line.

be $R = 10,000$. Along the training process, the standard deviation ϵ_r (see (39)) of the exploration noise is set to decay from the beginning $\boldsymbol{\epsilon}^{\max} = 0.15$ to $\boldsymbol{\epsilon}^{\min} = 0.075$ at the end of the training $R_{\boldsymbol{\epsilon}} = R = 10,000$.

We should also note that the entire Markov schedule decision process will be run through in each training epoch. In other words, given a realized demand profile, the transition experiences will be sampled and the parameters of the underlying actor and critic ANNs will be updated until all passengers in the system have been loaded and served. Depending on the magnitude and profile of the realized demand, it is observed from our experiment that around 60 - 100 train service runs will be dispatched by the schedule agent. Consequently, we could observe about 60 to 100 stage-to-stage transition samples in one single training epoch. This implies the transition sampling and the ANN parameter updating will be carried out 60 - 100 times in one single training epoch. With $R = 10,000$ training epochs, this would give approximately a total of 600,000 to 1,000,000 transition experience sampling and parameters updating throughout the entire training process. All computations are conducted on a Linux machine with Intel Core 2.70GHz CPU and 8 GB RAM. The related algorithms are coded in Python supported by the implementation of Pytorch (Paszke et al., 2017).

4.2. Training process of actor-critic schedule control agent

We start with presenting the performance of the DDPG training algorithm. The actor-critic schedule control agent is trained over the three predefined demand scenarios with different levels of uncertainty: D00, R25, and R50. The results are illustrated in Fig. 6. In the figure, the gray shadowed lines are the profiles of the objective function value (or the 'performance score', c_n) delivered by the schedule control agent at each training epoch under each demand scenario. The solid lines are the moving average (MA) values over 100 training epoch observations of the profiles. After around 1000 epochs, the profiles of the performance score are generally stabilized but exhibit fluctuations caused by the random decision exploration (with $\boldsymbol{\chi}$) and the background stochastic demand. The fluctuations due to exploration however would decrease in later training epochs with the decay of $\boldsymbol{\chi}$. We can observe that the magnitude of demand uncertainty has an impact on the

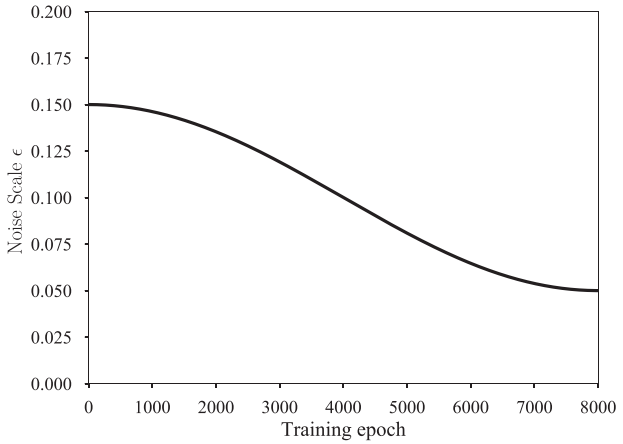


Fig. 5. The exploration noise (ϵ_r) profile adopted in the DDPG training process.

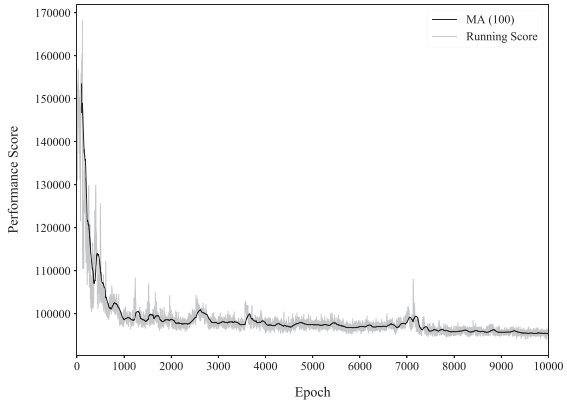
stability of the training process. The scheduling agent has more difficulty in delivering a stable performance in an environment with higher degree of uncertainty (say, in scenarios R25 and R50). Similar findings have indeed been found in related studies on stochastic control and robust optimization (Lo and Chow, 2002; Chow and Li, 2014; Li et al., 2019)).

To reveal the real performance of the schedule agent without being masked by the explorative noise $\boldsymbol{\chi}$ in the training process, an usual practice is to evaluate the performance of the schedule agent every 250 epochs (i.e. the 'test point') without adding explorative noise to its decisions. The performance statistics of the schedule agent is derived from 50 Monte Carlo runs for stochastic demand scenarios (R25 and R50). The evaluation results are shown in Fig. 7, where the black lines give the mean values of the performance; the gray areas (in Fig. 7a and c) represent the minimum and maximum ranges of the performance scores over the 50 Monte Carlo runs. The agent generally improves the mean performance score over the training process under all demand scenarios while maintaining the performance ranges steady over the imposed stochastic demand. This suggests the schedule agent is able to improve the overall system efficiency (in terms of reducing passenger and operation costs) and maintains the robustness. We should note that this is counter-intuitive because studies on robust design suggest that improvement in efficiency or robustness would typically come at the expense of the other (see Chow and Li, 2014; Li et al., 2019). The 'win-win' in both system efficiency and robustness delivered by the schedule agent suggests that the agent is able to learn to capture the sophisticated system dynamics driven by the stochastic demand and respond effectively to the stochastic variations through the underlying ANNs and stochastic batch optimization-based training process.

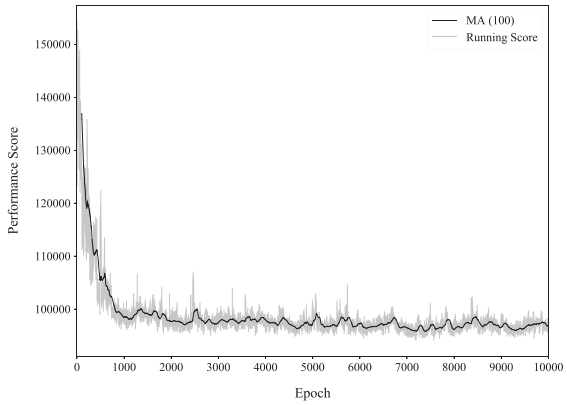
We would also like to further comment on the convergence of the training and testing processes shown herein. As discussed in Powell (2011), there is no reliable and implementable convergence or stopping rules for the agent training process shown herein. There are two reasons driving the structural fluctuations observed in the figures. One is due to the dynamic and stochastic passenger demand in the metro system which we consider, and the other is the incorporation of the explorative noise $\boldsymbol{\chi}$ during the training process. The effect of demand dynamics and stochasticity on the training and testing process can indeed be shown in the figures as discussed above. The training and testing process under deterministic demand scenario (DDPG-D00) is affected by the explorative noise $\boldsymbol{\chi}$ only. In fact, should one remove the explorative noise $\boldsymbol{\chi}$ in the deterministic demand scenario, the DDPG-D00 agent will then simply deliver a deterministic policy based on the prevailing settings of the actor ANN $\pi_{\boldsymbol{\phi}}$. With no exploration and demand variations, the DDPG-D00 agent will simply deliver the same solution $\bar{\boldsymbol{x}}_n$ every stage. As a result, the training and testing processes may converge fairly quickly, but only very likely to a bad solution without exploration in the process. To illustrate the service schedule derived by the actor-critic agent, Figs. 8 and 9 show respectively the train diagram and dispatching headways derived by an agent trained under the deterministic demand scenario (D00). Both figures illustrate the schedule agent could respond effectively to the demand surge by dispatching more trains during the period (e.g. 07:30-09:30) for reducing the passengers' waiting time, while reducing the number of service runs during the less busy period for saving the operating costs.

4.3. Benchmarking against established heuristics

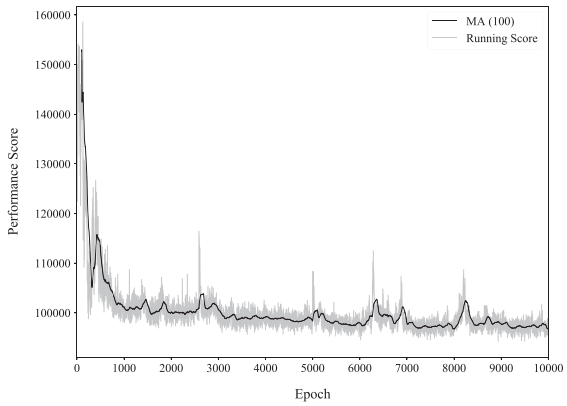
The DDPG training algorithm is now benchmarked against a selected set of well-established heuristics for service scheduling purpose. The demand scenario is set to be deterministic due to the consideration that these heuristics, as well as most existing methods, are designed as optimizers for deterministic scenarios (see e.g. Niu and Zhou, 2013; Sun et al., 2014; Liu et al., 2018). Based on our literature review (see Section 1), we select differential evolution (DE), particle swarm optimization (PSO), and genetic algorithm (GA) for the benchmarking purpose. For brevity, we refer the readers to Das and Suganthan (2011), Eberhart and Shi (2000), and Chow and Pavlides (2018), respectively for details of these algorithms. For the present study, we adopt an update generation of size 1500 and a population of size 50 for each of these heuristic algo-



(a) Scenario D00

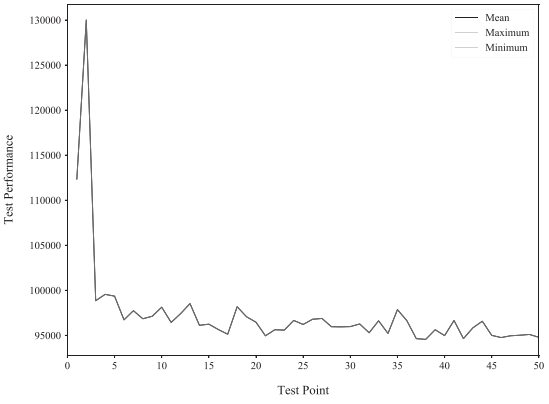


(b) Scenario R25

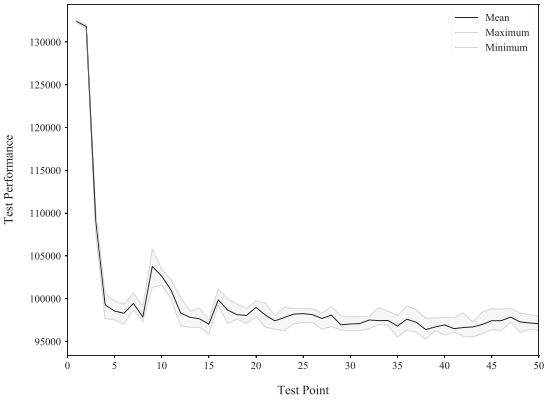


(c) Scenario R50

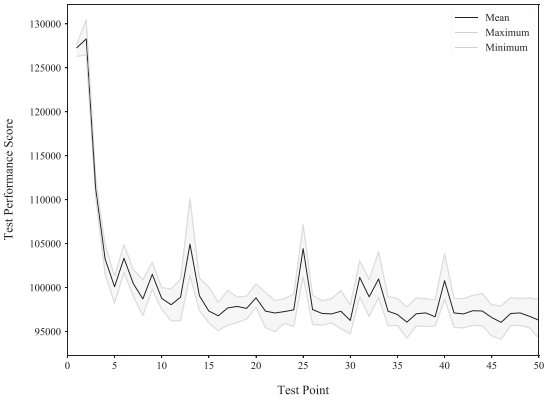
Fig. 6. Training process of the actor-critic schedule agent under different demand scenarios.



(a) Scenario D00



(b) Scenario R25



(c) Scenario R50

Fig. 7. Test of the actor-critic schedule agent during the training process.

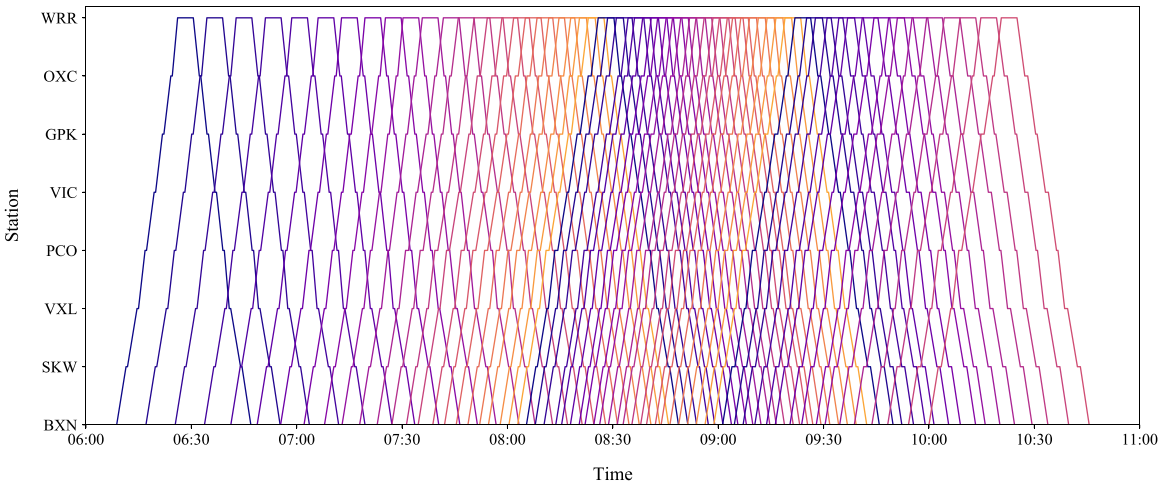


Fig. 8. Service runs scheduled by the actor-critic control agent (DDPG-D00) trained under the deterministic demand scenario.

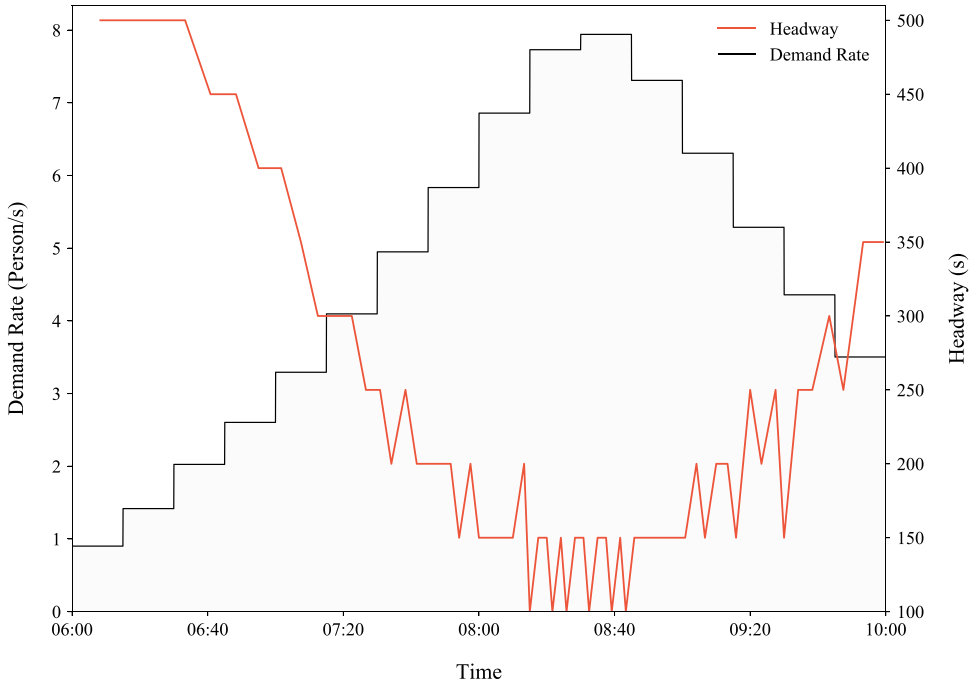


Fig. 9. Dispatching headways derived by the actor-critic control agent (DDPG-D00) trained under the deterministic demand scenario.

rithms. Following Das and Suganthan (2011), the scaling factor and crossover rate are set to be 0.6 and 0.9 respectively in the DE implementation. For GA, a crossover rate of 0.8 and a mutation rate of 0.05 are applied. Finally, following Eberhart and Shi (2000), our PSO implementation uses an inertia weight set as 0.729, and all acceleration constants are set to be 1.49445. Besides, the particle velocity is limited to the range [-0.20, 0.20].

Without the actor-critic framework (i.e. the ANNs), the metaheuristics search through directly the decision space for an optimal schedule that would give the minimum total cost over a predefined planning period. With the actor-critic framework, the schedule agent would use the ANNs (actor and critic), which represent respectively the policy function and the value function, to determine the optimal schedule policy. The aim of the DDPG training algorithm is to determine the parameters which characterize the actor and critic ANNs. The gives the different goals for the metaheuristics and the DDPG algorithm: the metaheuristics aims to seek the optimal schedule solutions directly, while the DDPG algorithm aims to seek the optimal ANN parameter sets, and hence the optimal schedule solutions via the ANNs under the actor-critic framework. From this perspective, the problems to be solved by the metaheuristics and DDPG are similar in terms of nature. Their difference hence will be dependent on their algorithmic performances.

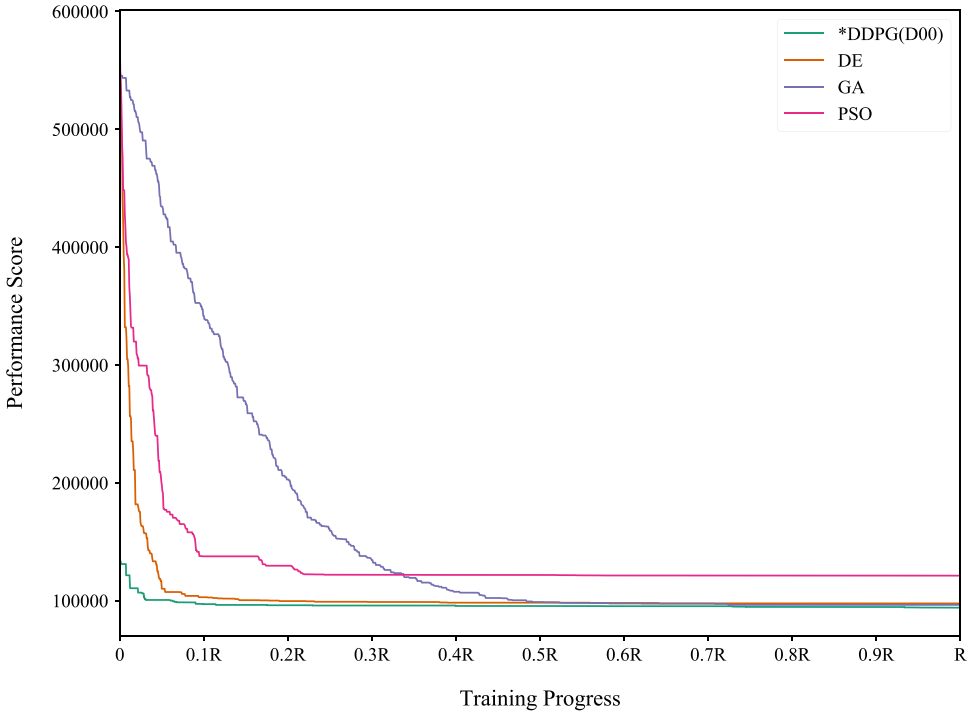


Fig. 10. Progression of DDPG algorithm and benchmarking heuristics under deterministic demand scenario.

The progression of the optimization or training algorithm is shown in Fig. 10. It is noted that different optimization or training algorithms would have different computing procedure and have different definition for what one iteration means. To standardize the comparison regardless of the differences in the computing procedures, the optimization progression is parameterized by the total number of training epochs or iterations R involved in each of the algorithms. Moreover, for comparison all algorithms are set to start with the same initial solution (Lo and Chow, 2002) which is associated with an initial performance score (cost) of 5.5×10^5. It is observed that all algorithms converge well and deliver generally similar performance under a deterministic demand setting. We also notice the DDPG training algorithm exhibits the greatest progress after one training epoch, where it brings the performance score down to 1.2×10^5.

The experiment results first support the argument that the DDPG algorithm is able to deliver a similar performance via the actor-critic framework that is comparable to the well-established metaheuristics. We also note that the DDPG algorithm is able to converge and reach the ultimate solution faster than the benchmarking heuristics. This is due to the use of efficient gradient-based solution algorithms (e.g. the Adam algorithm adopted herein) which are specifically designed for solving ANNs for training the schedule control agent. Such capability contrasts with the benchmarking metaheuristics which are mostly generic probabilistic population-based search. Finally, we should not be too surprised with the relatively inferior performance by PSO. Following general guidelines on parameter settings, the heuristics are only adopted herein as a benchmark measuring the performance of the DDPG training algorithm under an offline test case with deterministic demand. After all, the performance of any of these heuristics could be influenced by fine tuning the settings of the underlying parameters and the optimal parameter settings can be case dependent. There can only be general guidelines but no absolute rule on how these parameters should be optimally set.

Table 2 further summarizes the algorithms' performance in terms of computing times and objective function values delivered under the deterministic demand. Despite the fact that the performances are close, the table shows that the DDPG algorithm can indeed outperform the heuristics in terms of both performance (i.e. lower objective function values) and computing time. This highlights the capability of the underlying ANNs in the actor-critic deep reinforcement learning framework in capturing the complex and non-linear state-decision dynamics. Moreover, it is noted that the settings of the DDPG algorithm allows the actor-critic schedule agent to be trained under an environment with stochastic demand. Such capability cannot be supported by traditional heuristics including DE, GA, and PSO considered herein, and the significance of capturing stochastic demand will be further revealed in the following section of the paper.

For reference, Table 2 also includes the performance of the actor-critic agent trained through the stochastic demand scenarios (R25 and R50). Without disturbances caused by the stochastic demand variations, the agent trained by the deterministic demand (DDPG-D00) achieves the best performance. In contrast, the control agents (DDPG-R25 and DDPG-R50) trained by the stochastic demand scenarios tend to be more conservative. This can be exemplified by the fact that the two agents DDPG-R25 and DDPG-R50 decide to dispatch one additional service run in case there is an unexpected demand surge,

Table 2
Performance comparison of schedule optimizers under different training and optimization processes.

Method	Scenario	Optimal cost (£)	Passenger cost (£)	Operating cost (£)	Service run (runs)	Computing time (s)
DDPG	D00	**94036**	35,146	**152926**	**64**	6120
DDPG	R25	94,455	**34388**	154,523	65	6519
DDPG	R50	95,276	36,152	154,400	65	6875
GA	D00	96,405	38,043	154,768	**64**	8144
DE	D00	97,647	36,358	158,935	65	7177
PSO	D00	121,213	82,259	160,168	66	7093

Table 3
Performance of schedule optimizers under stochastic demand with 25% coefficient of variation.

Method	Mean total cost (£)	Mean passenger cost (£)	Mean penalty cost (£)	Mean operating cost (£)	Service run (runs)
DDPG-D00	95,518	36,898	1111	**152927**	**64.00**
DDPG-R25	**95267**	**34328**	**0**	156,206	65.79
DDPG-R50	95,384	36,226	**0**	154,442	65.02
GA	97,723	39,130	1547	154,770	**64.00**
DE	98,195	36,864	598	158,929	65.00
PSO	123,409	83,828	2825	160,165	66.00

Table 4
Performance of schedule optimizers under stochastic demand with 50% coefficient of variation.

Method	Mean total cost (£)	Mean passenger cost (£)	Mean penalty cost (£)	Mean operating cost (£)	Service run (runs)
DDPG-D00	102,835	41,218	11,473	**152978**	**64.00**
DDPG-R25	96,087	**34728**	**0**	157,446	66.34
DDPG-R50	**95965**	36,395	**0**	155,634	65.55
GA	103,625	43,164	9262	154,824	**64.00**
DE	100,514	38,855	3192	158,982	65.00
PSO	141,488	88,307	34,461	160,209	66.00

even though the extra train run induces a higher operating cost. Despite its better performance in the training process, the schedule agent DDPG-D00 can be less robust than its counterparts DDPG-R25 and DDPG-R50 when being applied to an environment with high level of stochasticity. Further discussion and illustration will be presented in the following section of the paper. Finally, we note the extra computing times spent in the training process under stochastic demand scenarios (R25 and R50) are due to the extra effort needed related to sampling of OD matrix settings from the distribution $\boldsymbol{O}_t$.

4.4. Testing under environment with stochastic demand

The trained actor-critic schedule agents are now tested under an environment driven by stochastic demand. This is to mimic situations when the trained schedule agents are being applied in real time to real world scenarios with imperfectly known demand settings (Lo and Chow, 2002; Chow and Li, 2014; Li et al., 2019). For benchmarking purpose, the trained schedule agents will be compared against fixed train service schedules derived by the meta-heuristics (DE, GA, and PSO) for the deterministic demand scenario. It is noted that in theory these meta-heuristics can also be used for real time applications, while it is subject to the condition that sufficient time is allowed for these meta-heuristics to compute updated schedules with respect to the prevailing system conditions. In contrast, the trained actor-critic control agent will be able to respond to prevailing demand variations without re-optimization or re-training. Nevertheless, the experiment herein merely aims to evaluate the performance of the adaptive actor-critic schedule agents with respect to fixed train schedules, instead of aiming to compare the performances of different optimization algorithms. Moreover, with the discrepancies between the demand profiles in the training process and the actual environment, there will be a possibility that some passengers may be left unserved toward the end of the service period due to underestimation of demand during the training stage. Following DfT (2014), we introduce a penalty of £16.15 for each residual passenger left by the end of the service for evaluation. The agents and schedules are tested over two stochastic demand scenarios with coefficients of variation of 25% and 50%. The performance statistics are derived from 100 Monte Carlo simulation runs and the results are summarized in Tables 3 and 4.

The results show that the overall performances of all schedule agents and fixed schedules generally deteriorate with increasing demand stochasticity. The adaptive actor-critic agents however can still outperform the fixed schedules derived from their benchmarking heuristics due to their responsiveness to prevailing demand variations. Despite having a lower operating cost with less service runs, the actor-critic agent DDPG-D00 is outperformed by its counterparts DDPG-R25 and DDPG-R50

Table 5
Performance comparison in a test scenario with shifted demand profile.

Method	Total cost (£)	Passenger cost (£)	Penalty cost (£)	Operating cost (£)	Fleet size (trains)
DDPG-R50	**95368**	36,320	0	154,415	65
DDPG-R25	95,476	**34254**	0	156,698	66
DDPG-D00	107,903	62,870	0	**152937**	**64**
DE	108,543	58,148	0	158,937	65
GA	111,937	69,106	0	154,769	64
PSO	137,162	114,132	0	160,192	66

in terms of passenger cost reduction due to the unexpected demand surges induced by the stochastic variations. The deterministic training environment causes the actor-critic control agent being over-fitted to the design demand and hence losing the robustness of coping with unexpected demand changes. In particular, there are respectively £1111 and £11,473 cost associated with residual passengers left in the system by DDPG-D00 due to its underestimation of the demand variations. In contrast, the actor-critic agents DDPG-R25 and DDPG-R50 are able to respond to the unexpected demand surges with their schedule buffer introduced during the training stage through the stochastic demand scenarios. To gain further insight, the results of the Monte Carlo experiments herein are visualized by the violin plots in Fig. 11. The plots show the means, quartiles, minimum and maximum values over the 100 Monte Carlo simulation runs as well as associated tallies as represented by the width. It supports that the performances of the actor-critic control agents trained under stochastic demand scenarios (DDPG-R25, DDPG-R50) are more robust in terms of performance deviations even though the solutions delivered by DDPG-D00 can achieve a lower average cost. The findings echo our previous related study in Chow and Li (2014) and Li et al. (2019) and all highlight the importance of robust design and considering stochasticity for real world applications.

4.5. Testing under environment with unexpected changes in demand profile

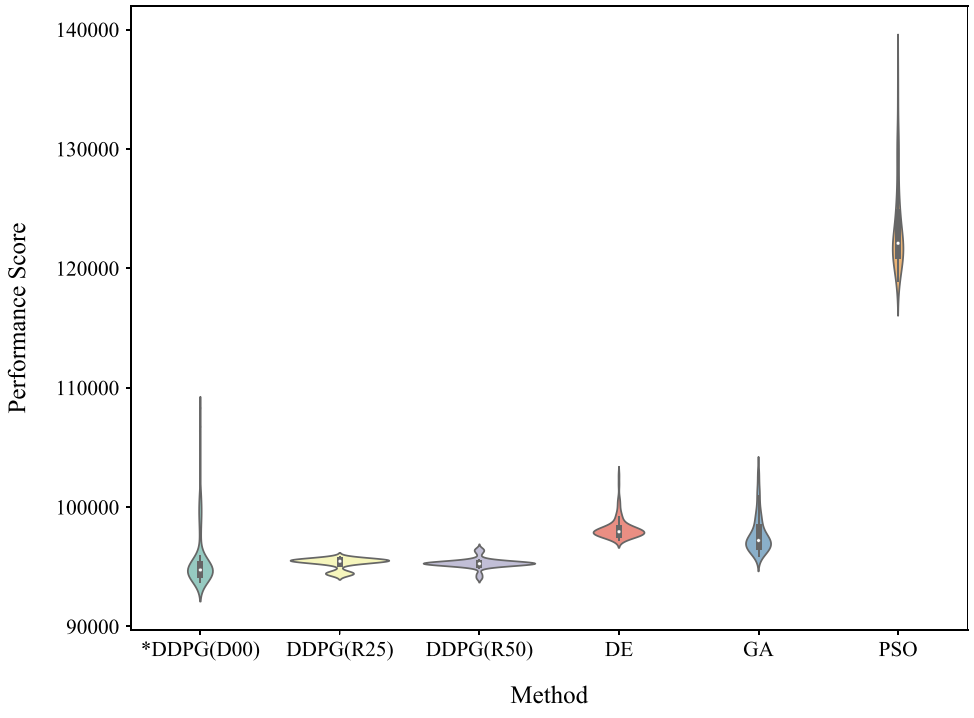
We further investigate the performance of the schedule agents under situation when there is a systematic discrepancy between the design demand profile and actual demand profile. As shown in Fig. 12, we consider there is an unexpected shift in demand profile in which there is a surge of rate of 1.0 person/s during 07:00-08:30 and then a drop in rate of 1.0 persons/s from 08:30 to 10:00. The performances of the schedule agents are summarized in Table 5. It is observed that the deterministic heuristics, and the schedule agent DDPG-D00, are affected significantly by this unexpected shift in demand profile. Nevertheless, the actor-critic agents trained with stochastic demand can deliver similar performance as in situations with no changes in demand distribution (see Tables 3 and 4) even without re-training. To gain further insight into the underlying schedule changes, the figure also shows the changes in dispatching headways induced by the actor-critic agent DDPG-R50 in response to the shifted demand. This experiment further reflects the system robustness gained by adopting the actor-critic schedule control agent with the underlying deep learning ANNs over fixed schedules computed by the benchmarking heuristics. We attribute the gained robustness in the actor-critic agent to the underlying ANNs which captures the sophisticated state-decision interaction and the associated stochasticity through its mathematical structure, and hence derives reliable estimates of states and corresponding schedule control with respect to prevailing demand variations in real time.

4.6. Pareto analysis

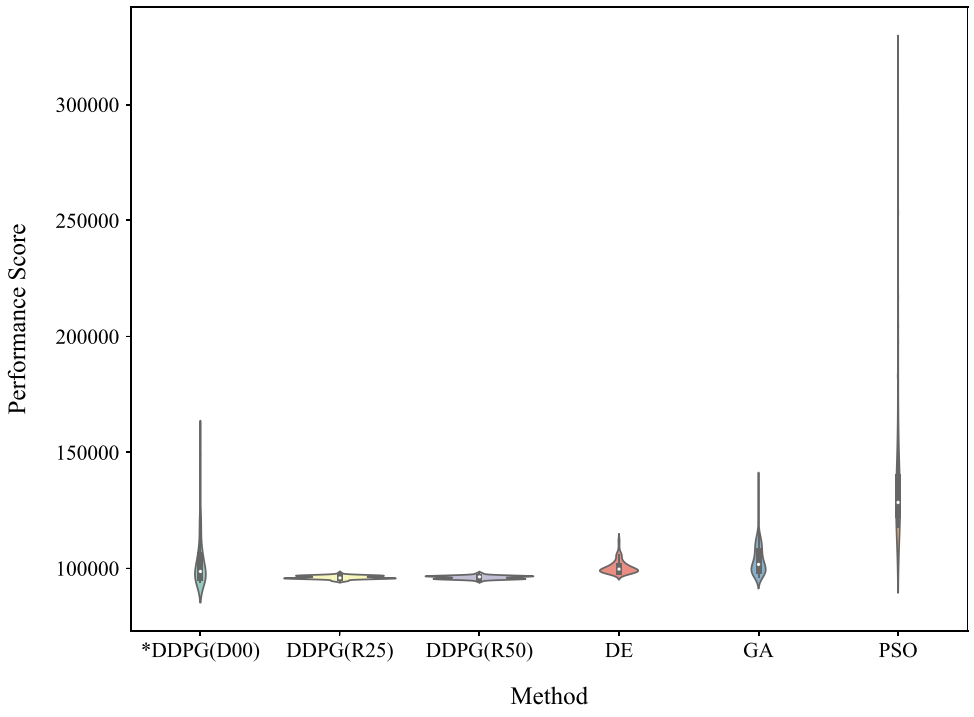
Finally, we conduct a Pareto analysis on the actor-critic reinforcement learning framework through setting different values for the trade-off coefficient α in the objective function. Such analysis is useful in multi-objective optimization as it reveals the sensitivity of different objectives with respect to changes in the trade-off parameter (Chow et al., 2017; Chow and Pavlides, 2018). In our study, we adopt seven values of α (= 0, 0.2, 0.4, 0.5, 0.6, 0.8, and 1.0) when training the DDPG-R25 actor-critic agent over the stochastic demand scenario. The performance of the agent is then evaluated over 100 Monte Carlo simulation runs driven by the stochastic demand with 25% coefficient of variation associated with the mean values.

Fig. 13 depicts the Pareto solutions in which we plot the logarithm of passenger waiting costs against the logarithm of the operating costs. Each data point plotted herein includes the minimum, mean, and maximum cost values obtained from the Monte Carlo simulation runs delivered by the DDPG-R25 agent. The data points are also illustrated with the associated number of service runs N dispatched with that specific setting of α. Referring to the objective function set in (18), a small value of α (e.g. $\alpha = 0$) implies more weight to be put on saving the operating cost, and hence less service runs on average (e.g. $N = 62.65$) will be dispatched. In contrast, a high value of α (e.g. $\alpha = 1$) would shift the focus toward passengers and hence the agent would dispatch more service runs on average ($N = 126$) to reduce the passengers' waiting times.

The figure shows that satisfactory performance in terms of minimizing both passengers' and operator's costs can be achieved by setting α in the interval [0.2, 0.6]. An increase or decrease of α beyond this range, say setting $\alpha = 0.0$ or $\alpha = 1.0$, could only gain little extra benefit on one of the objectives but at the great expense of the other. The pattern of the distribution of the Pareto solutions would be dependent on the scenarios such as the magnitude of the demand and



(a) Scenario R25



(b) Scenario R50

Fig. 11. Violin plot of performance distribution of the schedule optimizers under stochastic demand scenarios R25 and R50.

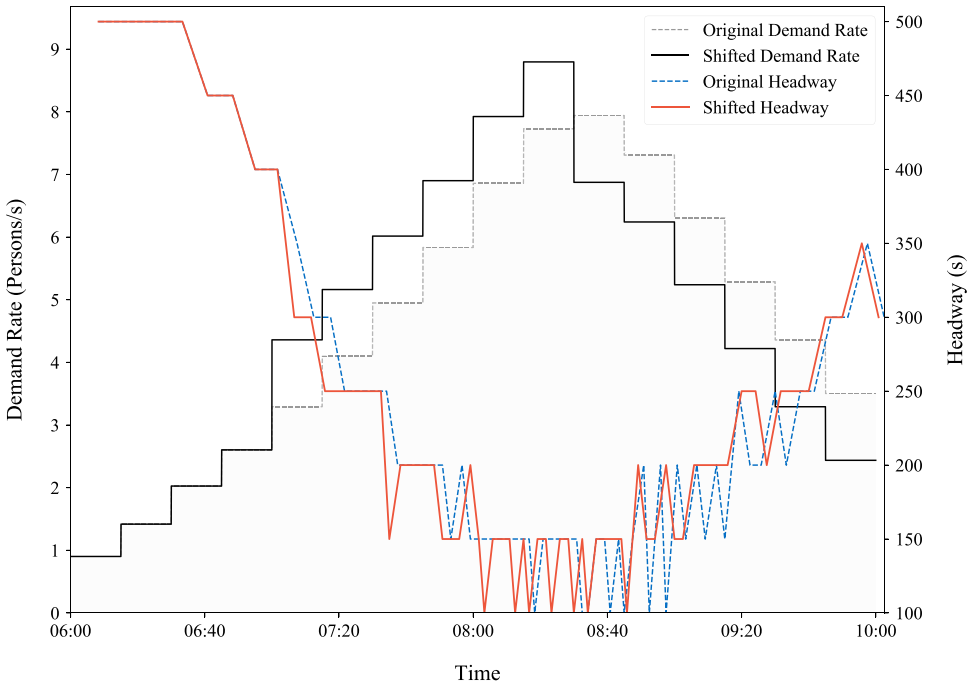


Fig. 12. Shifted demand profile and corresponding dispatching headways changes under actor-critic schedule control agent DDPG-R50.

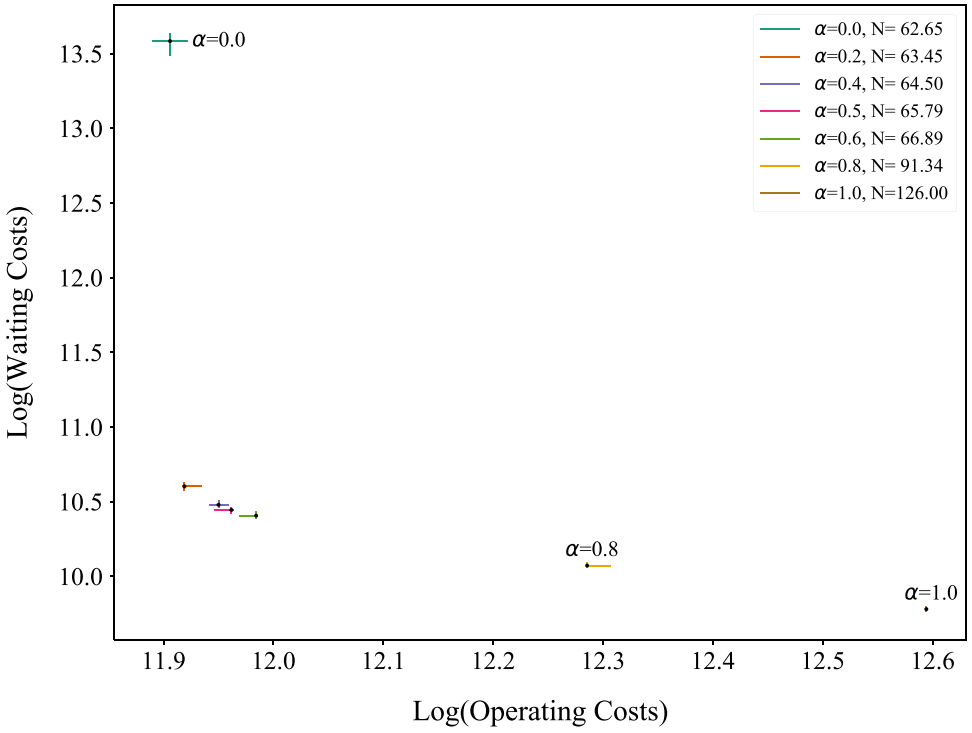


Fig. 13. Pareto solutions derived from the trained actor-critic schedule control agent DDPG-R25.

capacity of the rolling stock (Chow and Pavlides, 2018). However, for this specific application, the results demonstrate that $\alpha = 0.5$ is an effective setting in reducing both passengers' and operator's costs.

5. Conclusions

This study develops a novel metro service schedule controller for real time applications based on an actor-critic deep reinforcement learning framework. Driven by a stochastic demand profile, the schedule control agent derives a set of loop service runs with a fixed number of trains for an objective minimizing both passenger waiting times and train operating costs. The state and decision spaces are parameterized and represented respectively by a critic ANN and an actor ANN. We also present a deep deterministic policy gradient (DDPG) learning algorithm for training the schedule control agent before it can be applied for real use. Our experiment results with the London Victoria line scenario show that the actor-critic control agent can outperform the selected set of meta-heuristics (DE, PSO, and GA) over different demand scenarios in terms of both efficiency and robustness in both training and testing stages.

Deep reinforcement learning algorithms have been showing promising results in mimicking or even outperforming human experts in complicated tasks through various experiments, most famously exemplified by the Deepminds AlphaGo which conquered the world champions of the Go board game (Silver et al., 2016). This study is among the first which integrates this emerging and exciting technology in computer science into dynamic transit system optimization in the transportation field. With the advancements in data analytics and computing power, it offers great potential in improving the efficiency and robustness of our urban transport systems, which are crucial to sustainable economic and social development of future cities. Several prospective research directions have been identified. In addition to the demand management, we are extending our optimization framework to incident detection and management on the operational side. Moreover, the framework is only applied to a single metro service line. With the computational effectiveness shown, the present framework will be extended to network-wide and multi-modal management with multiple metro service lines and even multiple transport modes (Zhang and Lo, 2020). Capturing the transfer behavior of passengers and the propagation of disturbances along different transport services will be an interesting and challenging issue to address.

Acknowledgments

The authors would like to thank Prof. Robin Lindsey and the anonymous referees for their constructive comments. This study was supported by a Hong Kong Research Grant Council Theme-based Research Scheme (Ref: T32-101/15-R), the National Natural Science Foundation of China (Ref: 71971182), and an Applied Research Grant (Ref: 9667185) awarded by the City University of Hong Kong.

Appendix A. List of notation

Tables A.1 –A.3

Table A.1
Notation used in the stochastic transition model.

Notation	Definition
i, j	indices of station nodes.
n	index of service run (decision stage).
t	index of time interval.
$d_{i,i+1}$	distance between station nodes i and $i+1$.
$\mathcal{I}$	set of station nodes.
$\mathcal{I}_{\mathcal{M}}$	set of non-terminal station nodes.
$\mathcal{I}_{\mathcal{T}}$	set of terminal station nodes.
$(\Delta_{\mathcal{T}})_i$	shunting time at station node i.
K	fleet size of rolling stock.
N	nominal number of total services.
T	length of simulation period.
$\boldsymbol{O}_t$	$\boldsymbol{O}_t = [O_{ij}(t)]$, stochastic origin-destination demand matrix over time t.
$O_{ij}(t)$	demand rate between station pair (i, j) over time interval t.
$\hat{O}_{ij}(t)$	a specific realization of $O_{ij}(t)$.
$\sigma_{n,i}$	departure time of service n from station node i.
$\tau_{n,i}$	arrival time of service n at station node i.
$\Psi_i(t)$	cumulative arrival function of passengers at station i by time t.
$g_{n,i}$	cumulative number of passengers transport from station i by service run n.
Λ	train capacity.
$\lambda_{n+1,i(j)}$	number of passengers who want to board service $n+1$ at node i for node j.
$\hat{\lambda}_{n+1,i(j)}$	actual number of passengers who can board service $n+1$ at node i for node j.
$\hat{\lambda}_{n+1,i}$	total number of passengers who can actually board service $n+1$ at node i.
$\nu_{n+1,i}$	total number of passengers getting off from service $n+1$ at node i.
$\rho_{n+1,i}$	number of on-board passengers when the train service $n+1$ leaves station i.
$l_{n,i(j)}$	number of passengers left at station i wanting to go to station j by the departure of service n.
$l_{n,i}$	total number of passengers left at station i by the departure of service n.

Table A.2
Notation used in the Markov decision process.

Notation	Definition
$\boldsymbol{s}_{n+1} \sim P(\boldsymbol{s}_n, \boldsymbol{x}_n \vert \boldsymbol{O}_t)$	stochastic transition from stage n to $n+1$ driven by stochastic demand $\boldsymbol{O}_t$.
$\boldsymbol{s}_n$	$\boldsymbol{s}_n = (\tau_{n,i}, \sigma_{n,i}, l_{n,i})$, state tuple at stage n.
$\boldsymbol{x}_n$	$\boldsymbol{x}_n = (h_{n+1}, u_{n+1,i}, w_{n+1,i})$, decision tuple at stage n.
h_{n+1}	dispatching headway between service runs n and $n+1$ (decision variable).
$u_{n+1,i}$	running time of service $n+1$ over station pair i and $i+1$ (decision variable).
$w_{n+1,i}$	dwell time of service $n+1$ at station i (decision variable).
η_n	binary terminal indicator at stage n.
c_n	total cost function at stage n.
c_n^W	passenger waiting cost for service n.
c_n^V	operational cost for service n.
ζ^D	unit operational cost factor associated with running distance.
ζ^F	unit operational cost factor associated with energy consumption.
ζ^T	unit operational cost factor associated with running time.
ζ^W	unit monetary value of passenger waiting time.
β_1, β_2, β_3	coefficients of the Davis formula (energy consumption).
$v_{n+1,i}$	running speed of service n over track section between nodes i and $i+1$.
F	resistance force.
M_T	tare mass of a train-set.
M_P	average weight of a passenger with personal belongings.
α	trade-off coefficient between perspectives of passengers' and operators' in c_n.
$h_{\min}$, $h_{\max}$	minimum and maximum dispatching headways.
$w_{\min}$, $w_{\max}$	minimum and maximum train dwell times.
$(u_i)_{\min}$, $(u_i)_{\max}$	minimum and maximum section running time between nodes i and $i+1$.
Δ_S	required safety margin between consecutive services.

Table A.3
Notation used in the actor-critic reinforcement learning framework.

Notation	Definition
$(x_{(n)})_{\min}$, $(x_{(n)})_{\max}$	stage-dependent boundary defined by constraints.
B	training batch size in the DDPG solution process.
$Q_{\boldsymbol{\theta}}$	critic artificial neural network (ANN) with parameters $\boldsymbol{\theta}$.
R	number of training epochs.
R_ϵ	number of epochs for ϵ_r.
V	state value function.
Z	size of experience replay buffer.
Ω_n	stage dependent decision space of $\boldsymbol{x}_n$ at decision stage n.
$\tilde{\boldsymbol{x}}_n$	decision embedding tuple at decision stage n.
$\boldsymbol{\chi}$	exploration noise.
$\boldsymbol{\epsilon}^{\min}$, $\boldsymbol{\epsilon}^{\max}$	minimum and maximum standard deviation of exploration noise.
$\boldsymbol{\phi}$, $\boldsymbol{\phi}'$	parameter vectors of actor and target actor ANNs.
$\boldsymbol{\theta}$, $\boldsymbol{\theta}'$	parameter vector of critic and target critic ANNs.
ϵ_r	standard deviation of exploration noise.
γ	discount factor.
$\mathcal{L}_A(\boldsymbol{\phi})$	loss function to be minimized for updating actor ANN parameter.
$\mathcal{L}_Q(\boldsymbol{\theta})$	loss function to be minimized for updating critic ANN parameter.
$\pi_{\boldsymbol{\phi}}$	actor ANN.
ξ	smoothing factor for updating parameters.
b	index of transition experiences in a mini-batch.
f	a linear mapping for decoding decision embedding tuples.
r	index of training epochs.
y_b	target value for updating critic ANN parameter.

References

Alom, M., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M., Hasan, M., Van Essen, B., Awwal, A., Asari, V., 2019. A state-of-the-art survey on deep learning theory and architectures. Electronics 8 (3), 292. doi:10.3390/electronics8030292.

Barrena, E., Canca, D., Coelho, L., Laporte, G., 2014. Exact formulations and algorithm for the train timetabling problem with dynamic demand. Comput. Oper. Res. 44, 66–74. doi:10.1016/j.cor.2013.11.003.

Bertsekas, D., 2019. Reinforcement Learning and Optimal Control. Athena Scientific, Belmont, M.A..

Cacchiani, V., Furini, F., Kidd, M., 2016. Approaches to a real-world train timetabling problem in a railway node. Omega 58, 97–110. doi:10.1016/j.omega.2015.04.006.

Chang, C.S., Kwan, C.M., 2004. Evaluation of evolutionary algorithms for multi-objective train schedule optimization. In: Webb, G., Yu, X. (Eds.), AI 2004: Advances in Artificial Intelligence, pp. 803–815.

Chow, A.H.F., Li, S., Zhong, R., 2017. Multi-objective optimal control formulations for bus service reliability with traffic signals. Transp. Res. Part B 103, 248–268. doi:10.1016/j.trb.2017.02.006.

Chow, A.H.F., Li, Y., 2014. Robust optimization of dynamic motorway traffic via ramp metering. IEEE Trans. Intell. Transp. Syst. 15 (3), 1374–1380.

Chow, A.H.F., Pavlides, A., 2018. Cost functions and multi-objective timetabling of mixed train services. Transp. Res. Part A 113, 335–356. doi:10.1016/j.tra.2018.04.027.

Corman, F., D'Ariano, A., Marra, A., Pacciarelli, D., Sama, M., 2017. Integrating train scheduling and delay management in real-time railway traffic control. Transp. Res. Part E 105, 213–239.

Daganzo, C.F., Ouyang, Y., 2019. Public Transportation Systems:Principles of System Design, Operations Planning and Real-Time Control. World Scientific, Hackensack, N.J..
Das, S., Suganthan, P., 2011. Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15 (1), 4–31. doi:10.1109/TEVC.2010.2059031.
Davis, W., 1926. The tractive resistance of electric locomotives and cars. Gen. Electric Rev. 29 (10), 685–708.
DfT, 2014. TAG Unit 3.5.6 Values of Time and Vehicle Operating Costs, Transport analysis guidance data book. Technical Report. Department for Transport, United Kingdom.
D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains in a railway network. Eur. J. Oper. Res. 183 (2), 643–657. doi:10.1016/j.ejor.2006.10.034.
Eberhart, R.C., Shi, Y., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), vol. 1, pp. 84–88. doi:10.1109/CEC.2000.870279. PSO weights
Ghasempour, T., Heydecker, B.G., 2020. Adaptive railway traffic control using approximate dynamic programming. Transp. Res. Part C 113, 91–107.
Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks, 8Xavier init.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT press, MA.
Guo, X., Sun, H., Wu, J., Jin, J., Zhou, J., Gao, Z., 2017. Multiperiod-based timetable optimization for metro transit networks. Transp. Res. Part B 96, 46–67.
Hansen, H., Nawaz, M., Olsson, N., 2017. Using operational data to estimate the running resistance of trains. Estimation of the resistance in a set of Norwegian tunnels. J. Rail Transp. Plann. Manage. 7 (1), 62–76. doi:10.1016/j.jrtpm.2017.01.002.
Kingma, D. P., Ba, J., 2014. Adam: a method for stochastic optimization. arXiv:1412.6980[cs] Adam.
Li, X., Lo, H., 2014. An energy-efficient scheduling and speed control approach for metro rail operations. Transp. Res. Part B 64, 73–89.
Li, X., Lo, H., 2014. Energy minimization in dynamic train scheduling and control for metro rail operations. Transp. Res. Part B 70, 269–284.
Li, X., Wang, D., Li, K., Gao, Z., 2013. A green train scheduling model and fuzzy multi-objective optimization algorithm. Appl. Math. Model. 37 (4), 2063–2073.
Li, Y., 2017. Deep reinforcement learning: an overview. arXiv:1701.07274[cs.LG].
Li, Y., Chow, A.H.F., Zhong, R., 2019. Control strategies for dynamic motorway traffic subject to flow uncertainties. Transportmetrica B 7 (1), 559–575.
Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv:1509.02971[cs.LG].
Liu, R., Li, S., Yang, L., Yin, J., 2018. Energy-efficient subway train scheduling design with time-dependent demand based on an approximate dynamic programming approach. IEEE Trans. Syst. Man Cybern. 1–16. doi:10.1109/TSMC.2018.2818263.
Lo, H., Chow, A.H.F., 2002. Adaptive traffic control system: control strategy, prediction resolution, and accuracy. J. Adv. Transp. 36 (3), 323–347.
Loshchilov, I., Hutter, F., 2016. SGDR: Stochastic gradient descent with warm restarts. arXiv:1608.03983[cs, math].
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533. doi:10.1038/nature14236. DQN Nature
Mo, P., Yang, L., Wang, Y., Qi, J., 2019. A flexible metro train scheduling approach to minimize energy cost and passenger waiting time. Comput. Ind. Eng. 132, 412–432. doi:10.1016/j.cie.2019.04.031.
Newell, G., Potts, R., 1964. Maintaining a bus schedule. In: Proceedings of the 2nd Australian Road Research Board, 2, pp. 388–393.
Niu, H., Zhou, X., 2013. Optimizing urban rail timetable under time-dependent demand and oversaturated conditions. Transp. Res. Part C 36, 212–230. doi:10.1016/j.trc.2013.08.016.
Noursalehi, P., Koutsopoulos, H.N., Zhao, J., 2018. Real time transit demand prediction capturing station interactions and impact of special events. Transp. Res. Part C 97, 277–300.
Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic differentiation in PyTorch. Advances in Neural Information Processing Systems 30 (NIPS 2017), Pytorch.
Powell, W.B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality. John Wiley & Sons, N.J..
Robenek, T., Maknoon, Y., Azadeh, S., Chen, J., Bierlaire, M., 2016. Passenger centric train timetabling problem. Transp. Res. Part B 89, 107–126. doi:10.1016/j.trb.2016.04.003.
Sama, M., D'Ariano, A., Corman, F., Pacciarelli, D., 2017. A variable neighborhood search for fast train scheduling and routing during disturbed railway traffic situations. Comput. Oper. Res. 78, 480–499.
Scheepmaker, G., Goverde, R.M.P., Kroon, L., 2017. Review of energy-efficient train control and timetabling. Eur. J. Oper. Res. 257 (2), 355–376. doi:10.1016/j.ejor.2016.09.044.
Shao, H., Lam, W.H.K., Sumalee, A., Chen, A., Hazelton, M.L., 2014. Estimation of mean and covariance of peak hour origin destination demands from day-to-day traffic counts. Transp. Res. Part B 68, 52–75. doi:10.1016/j.trb.2014.06.002.
Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of go with deep neural networks and tree search. Nature 529 (7587), 484–489.
Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms. In: Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML), DPG.
Su, Z., Chow, A.H.F., Zheng, N., Huang, Y., Liang, E., Zhong, R., 2020. Neuro-dynamic programming for optimal control of macroscopic fundamental diagram systems. Transp. Res. Part C 116, #102628.
Sun, L., Jin, J., Lee, D.H., Axhausen, K.W., Erath, A., 2014. Demand-driven timetable design for metro services. Transp. Res. Part C 46, 284–299. doi:10.1016/j.trc.2014.06.003.
Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An introduction. MIT press, M.A..
Tampére, C.M.J., Corthout, R., Cattrysse, D., Immers, L., 2011. A generic class of first order node models for dynamic macroscopic simulation of traffic flows. Transp. Res. Part B 45, 289–309.
Umiliacchi, S., Nicholson, G., Zhao, N., Schmid, F., Roberts, C., 2016. Delay management and energy consumption minimisation on a single-track railway. IET Intel. Transp. Syst. 10, 73–89.
Yang, L., Qi, J., Li, S., Gao, Y., 2016. Collaborative optimization for train scheduling and train stop planning on high-speed railways. Omega 64, 57–76. doi:10.1016/j.omega.2015.11.003.
Yang, X., Chen, A., Ning, B., Tang, T., 2017. Bi-objective programming approach for solving the metro timetable optimization problem with dwell time uncertainty. Transp. Res. Part E 97, 22–37. doi:10.1016/j.tre.2016.10.012.
Yang, X., Li, X., Ning, B., Tang, T., 2016. A survey on energy-efficient train operation for urban rail transit. IEEE Trans. Intell. Transp. Syst. 17 (1), 2–13. doi:10.1109/TITS.2015.2447507.
Yang, Y., Fan, Y., Wets, R., 2018. Stochastic travel demand estimation: improving network identifiability using multi-day observation sets. Transp. Res. Part B 107, 192–211. doi:10.1016/j.trb.2017.10.007.
Yin, J., Chen, D., Li, L., 2014. Intelligent train operation algorithms for subway by expert system and reinforcement learning. IEEE Trans. Intell. Transp. Syst. 15 (6), 2561–2571.
Yin, J., Chen, D., Li, Y., 2016. Smart train operation algorithms based on expert knowledge andensemble CART for the electric locomotive. Knowl. Based Syst. 92, 78–91.
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 91, 178–210. doi:10.1016/j.trb.2016.05.009.
Zhang, S., Lo, H., 2020. Metro disruption management: contracting substitute bus service under uncertain system recovery time. Transp. Res. Part C 110, 98–122.
Zhang, T., Li, D., Qiao, Y., 2018. Comprehensive optimization of urban rail transit timetable by minimizing total travel times under time-dependent passenger demand and congested conditions. Appl. Math. Model. 58, 421–446. doi:10.1016/j.apm.2018.02.013.
Zhou, M., Wang, D., Li, Q., Yue, Y., Tu, W., Cao, R., 2017. Impacts of weather on public transport ridership: results from mining data from different sources. Transp. Res. Part C 75, 17–29. doi:10.1016/j.trc.2016.12.001.