Transportation Research Part C 119 (2020) 102715

Contents lists available at ScienceDirect

Transportation Research Part C

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

Deep reinforcement learning algorithm for dynamic pricing of express lanes with multiple access locations

Venktesh Pandey*, Evana Wang, Stephen D. Boyles

Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin, Austin, TX 78712, USA

ARTICLE INFO

Keywords:
Managed lanes
Express lanes
High occupancy/toll (HOT) lanes
Dynamic pricing
Deep reinforcement learning
Traffic control
Feedback control heuristic

ABSTRACT

This article develops a deep reinforcement learning (Deep-RL) framework for dynamic pricing on managed lanes with multiple access locations and heterogeneity in travelers' value of time, origin, and destination. This framework relaxes assumptions in the literature by considering multiple origins and destinations, multiple access locations to the managed lane, *en route* diversion of travelers, partial observability of the sensor readings, and stochastic demand and observations. The problem is formulated as a partially observable Markov decision process (POMDP) and policy gradient methods are used to determine tolls as a function of real-time observations. Tolls are modeled as continuous and stochastic variables and are determined using a feedforward neural network. The method is compared against a feedback control method used for dynamic pricing. We show that Deep-RL is effective in learning toll policies for maximizing revenue, minimizing total system travel time, and other joint weighted objectives, when tested on real-world transportation networks. The Deep-RL toll policies outperform the feedback control heuristic for the revenue maximization objective by generating revenues up to 8.5% higher than the heuristic and for the objective minimizing total system travel time (TSTT) by generating TSTT up to 8.4% lower than the heuristic. We also propose reward shaping methods for the POMDP to overcome the undesired behavior of toll policies, like the *jam-and-harvest* behavior of revenue-maximizing policies. Additionally, we test transferability of the algorithm trained on one set of inputs for new input distributions and offer recommendations on real-time implementations of Deep-RL algorithms. The source code for our experiments is available online at https://github.com/venktesh22/ExpressLanes_Deep-RL.

1. Introduction

1.1. Background and motivation

Priced managed lanes (MLs) are increasingly being used by many cities to mitigate traffic congestion and provide reliable travel time. As of January 2019, there are 41 managed lane projects across the United States (n.a., 2020). On these lanes, travelers pay a toll which changes with the time of day, or dynamically based on the congestion pattern, to experience less congested travel time from their origin to their destination. In recent years, managed lane networks have become increasingly complex, spanning longer corridors and having multiple entrance and exit locations. For example, the LBJ TEXpress lanes in Dallas, TX have 17 entrance ramps and 18 exit ramps, and three tolling segments with dynamics tolls (LBJ, 2016).

* Corresponding author.
E-mail address: venktesh@utexas.edu (V. Pandey).

https://doi.org/10.1016/j.trc.2020.102715
Received 22 January 2020; Received in revised form 5 June 2020; Accepted 7 July 2020
Available online 08 August 2020
0968-090X/

Dynamic pricing for MLs with multiple access locations is a complex control problem due to the heterogeneity in lane choice behavior of travelers with varying values of time and destinations of travel. Predicting driver behavior with certainty is difficult. A recent study showed that a binary logit model, commonly used for modeling lane choice, is inadequate in predicting heterogeneity in lane choice observations (Burris and Brady, 2018).

Several dynamic pricing algorithms have been explored in the literature that optimize tolls under varying assumptions on driver behavior. These include methods using stochastic dynamic programming (Yang et al., 2012), hybrid model predictive control (MPC) (Tan and Gao, 2018; Toledo et al., 2015), reinforcement learning (RL) (Zhu and Ukkusuri, 2015; Pandey and Boyles, 2018b), and approximate dynamic programming (Pandey and Boyles, 2018a). While these algorithms do well against existing heuristics, they make some or all of the following restricting assumptions, which we relax:

1. Restricted access for travelers: travelers do not exit the managed lane once they enter till their exit is reached (Yang et al., 2012; Zhu and Ukkusuri, 2015), and that only the first entry location is considered for lane-choice decision (Tan and Gao, 2018)
2. Fully observable system: toll operators have access to measurements of traffic density throughout the network for optimizing tolls (Yang et al., 2012; Tan and Gao, 2018; Zhu and Ukkusuri, 2015; Pandey and Boyles, 2018a,b)
3. Ignored traveler heterogeneity: a single vehicle class is considered with a single origin and destination (Yang et al., 2012; Zhu and Ukkusuri, 2015; Pandey and Boyles, 2018a)
4. Simplified traffic dynamics: for example, the flow dynamics on general-purpose lanes (GPLs) are assumed independent of vehicles using the ML (Yang et al., 2012); or the proportion of flow split at diverge points is assumed identical for all origins (Tan and Gao, 2018)

In addition, there are relatively few analyses on the conflict between optimization of multiple objectives with realistic constraints. Pandey and Boyles (2018a) showed that the revenue-maximizing tolls exhibit a *jam-and-harvest* (JAH) nature where GPLs are intentionally jammed to congestion earlier in the simulation to harvest more revenue towards the end. Handling such undesirable behavior of optimal policies has not been studied in the literature.

Furthermore, practical applicability of these algorithms in real-world environments is a less-explored question. Algorithms that optimize prices using a simulation model can be applied in real time using lookup tables. However, the transferability analysis of such lookup tables to new input distributions is not considered (Yang et al., 2012; Zhu and Ukkusuri, 2015; Pandey and Boyles, 2018a). The hybrid MPC algorithm in Tan and Gao (2018) follows a different procedure for practical applications. It predicts boundary traffic as an exogenous input using a simulation model and optimizes tolls over a finite horizon using real-time measurements of traffic densities and queue lengths. However, solving an MPC-based model with heterogeneous vehicle classes and partial observability of the system, without the restricting assumptions stated earlier, is complex and not fully studied. We thus require scalable algorithms for real-world networks that relax the assumptions on driver behavior and traffic flow, and transfer well from simulation settings to new input distributions.

In this article, we focus on pricing algorithms that rely on real-time density observations using sensors (such as loop detectors) located only at certain locations around the network without access to any information about the demand distribution or driver characteristics like the value of time (VOT) distribution. We use deep reinforcement learning (Deep-RL) algorithms for optimizing tolls while relaxing simplifying assumptions in the earlier literature. In the recent years, Deep-RL algorithms have been successfully used for applications such as playing Atari games and planning the motion of humanoid robots like `MuJoCo` (Arulkumaran et al., 2017). Similar algorithms have been applied for traffic signal control (Shabestary and Abdulhai, 2018), active traffic management (Belletti et al., 2017), and control of autonomous vehicles in mixed autonomy (Wu et al., 2017).

Simply applying Deep-RL as a "black box" is unlikely to yield effective solutions for pricing dynamic lanes, due to the size of the state space and the potential for undesirable jam-and-harvest behavior. We have introduced two domain-specific elements into our formulation and experiments: the use of a "decision route model" as a concise (polynomial-space) way of simulating route choice in corridors with multiple ML entrances and exits; and the use of reward shaping to avoid JAH phenomena. In addition, using Deep-RL algorithms, we relax assumptions in the literature by considering multiple origins and destinations, multiple access points to the managed lane facility, *en route* diversion of vehicles at each diverge point, and partial observability of traffic state. The key contributions of this article are:

- We demonstrate the usefulness of Deep-RL algorithms for solving dynamic pricing control problem under partial observability, and show that it performs well against existing heuristics, without requiring restricting assumptions on driver behavior or traffic dynamics.
- We apply multi-objective optimization methods for joint optimization of multiple objectives and overcome undesirable JAH characteristics of revenue-maximizing optimal policies.
- We conduct tests to verify the transferability of learned Deep-RL algorithms to new input distributions and model parameters, highlighting the critical role of lane choice models to ensure transferability. Using these experiments, we also make recommendations on real-time implementation of the algorithm.
- We develop an open-source framework for dynamic pricing using multiclass cell transmission model available for benchmarking future dynamic pricing experiments.

1.2. Related work

Many control problems have been studied in the area of transportation engineering including active traffic management strategies such as ramp metering, variable speed limits, dynamic lane use control, and adaptive traffic signal control (ATSC). These control problems can be broadly solved using three methods: open-loop optimal control methods (that solve the optimal control problem without incorporating real-time measurements), closed-loop control methods like MPC (that incorporate the feedback of real-time measurements and optimize over a rolling horizon), and lately RL methods where the optimal control is learned with an iterative interaction with the environment, possibly in simulated offline settings which can then be translated in real world settings. Refer Ferrara et al. (2018, Chapter 8), for an overview of control problems in the transportation domain.

The managed lane pricing problem is also a traffic control problem, where the chosen control directly impacts the driver behavior and thus the congestion pattern. There are three component models to the ML pricing problem (Gardner et al., 2013): a *lane choice model* that determines how travelers choose a lane given the tolls and travel times, a *traffic flow model* that models the interaction of vehicles in simulated environments, and a *toll pricing model* which determines the toll pricing objectives and how the optimization problem is solved to achieve the best value of the objective. Pandey (2016) presented a tabular comparison of component models for the existing models in the literature. In this research, we focus on the *toll pricing models*.

Toll pricing models for MLs with a single access point are commonly studied. Gardner et al. (2013) argued that for MLs with a single entrance and exit, the tolls minimizing the total system travel time (TSTT) also utilize the managed lanes to full capacity at all times. The authors developed an analytical formulation for tolls minimizing TSTT which send as many vehicles to the ML at each time step as is the capacity of the lane. Lou et al. (2011) used a self-learning approach for optimizing toll prices where the average VOT values were learnt using real-time measurements. Toledo et al. (2015) used a rolling horizon approach to optimize future tolls with predicted demand from traffic simulation; however, the method of exhaustive search to solve the non-convex control problem does not scale well for large managed lane networks.

For managed lanes with multiple access points, Tan and Gao (2018) presented a formulation where the proportion of vehicles entering the managed lane is optimized instead of directly optimizing the toll prices. The authors showed a one-to-one mapping between optimal toll prices and the proportion values, and transformed the control problem into a mixed-integer linear program which can be solved efficiently for networks with multiple access points. Dorogush and Kurzhanskiy (2015) used a similar method and optimized split ratios at each diverge, which are then used to determine toll prices; however, their analysis ignored the variation of incoming flow at each diverge. Apart from these optimal control based methods, Zhu and Ukkusuri (2015) and Pandey and Boyles (2018a) used RL methods, where the control problem is formulated as a Markov decision process (MDP) and the value function (or its equivalent Q-function) is learned by iterative interactions with the environment. However, the tests are conducted for discrete state and action spaces assuming full observability of the system. The present article is guided by advances in RL methods, and improves these earlier RL-based approaches for dynamic pricing.

Deep-RL improves traditional RL by using deep neural networks as function approximators, which has been effective in various control problems. See Arulkumaran et al. (2017) for a survey of Deep-RL applications. Application of Deep-RL algorithms for traffic control problems is not new. Belletti et al. (2017) developed an “expert-level” control of coordinated ramp metering using Deep-RL methods with multiple agents and achieved precise adaptive metering without requiring model calibration that does better than the traditional benchmark algorithm named ALINEA. Wu et al. (2017) used Deep-RL algorithms to solve the control problem of selecting the acceleration and brake of multiple autonomous vehicles (AVs) under conditions of mixed human vehicles and AVs to mitigate traffic congestion. When compared against classical approaches, their approach generated 10–20% lower TSTT. Other applications of Deep-RL algorithms are in the domain of ATSC including traditional one signal control (Genders and Razavi, 2016; Shabestary and Abdulhai, 2018), coordinated control of traffic signals (van der Pol, 2016), and large-scale multiagent control using Deep-RL methods (Chu et al., 2019). See Yau et al. (2017) for a review of RL algorithms in the area of ATSC.

The rest of the paper is organized as follows. Section 2 introduces the notation and presents the details of the model. Section 3 explains the Deep-RL algorithms and the feedback control heuristic against which the algorithm is compared. Section 4 presents the experimental analysis of Deep-RL algorithms on four test networks and discusses transferability analysis, multi-objective optimization, and compares performance of Deep-RL algorithms against other algorithms from the literature. Section 5 concludes the paper and suggests topics for future work.

2. Model for deep reinforcement learning

2.1. Network notation

Consider the directed network shown in Fig. 1 which is an abstraction of a managed lane network. The upper set of links form MLs, the lower set of links form GPLs, and the ramps connect the two lanes at various access points. As we describe the network, we label the assumptions made in our model as “A#”. We also label ideas for future work as “FW#”.

Let N represent the set of all nodes and $A = \{(i, j)|i, j \in N\}$ represent the set of all links in the network. Let N_o denote the set of all origins and N_d denote the set of all destinations. We assume that origins and destinations connect to the network through nodes on the GPLs (A#1) and the only way to access the MLs is through on-ramps leading towards the lane. This is a reasonable assumption as most current ML installations allow access to MLs only through ramps from the GPL. If there is a direct access to the ML from outside the network, the current framework can still be used by appropriately adjusting the lane choice model explained in Section 2.2.

The time horizon is divided into equal time steps, each Δt units long. The set of all time periods is given by $\mathscr{T} = \{t_0, t_1, t_2, ..., t_{T/\Delta t}\}$,

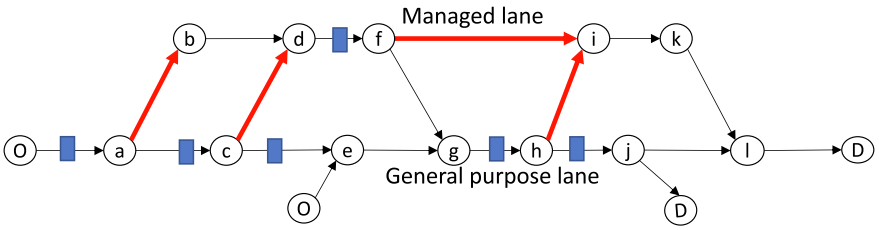


Fig. 1. Managed lane network with multiple entrances and exits where links with higher thickness are tolled, and links with a box are observed by the toll operator.

where T, an integral multiple of Δt, is the time horizon. Tolls are updated after every $\Delta\tau = m\Delta t$ time units, where m is a positive integer fixed by the tolling agency. Define $\mathscr{T}_\tau = \{k | t_{km} \in \mathscr{T},$ where $k \in \{0, 1, 2, ...\}\}$ as the set of time periods where tolls are updated, indexed in increasing order of positive integers. Then, $|\mathscr{T}_\tau| = T/\Delta\tau + 1$. For example, Fig. 2 shows different elements of time where $m = 4$ and $T = 16\Delta t$. For the figure, $\mathscr{T} = \{t_0, t_1, t_2, ..., t_{16}\}$ and $\mathscr{T}_\tau = \{0, 1, 2, 3, 4\}$.

The demand between an origin and a destination is a random variable. A toll operator does not know the demand distribution, but only relies on the observed realizations of demand. However, for simulation purposes, we model the demand of vehicles from origin $r \in N_o$ to destination $s \in N_d$ at time $t \in \mathscr{T}$ to be a rectified Gaussian random variable with mean $d_{rs}(t)$ and standard deviation σ_d, and ignore correlations of demand between different origin–destination (OD) pairs and across time. The mean demand $d_{rs}(t)$ can be estimated by observing the historical data of the managed lane facility or from the regional model.

Let V denote the set of all values of VOT (assumed to be a discrete distribution for the population, A#2) and p_v be the proportion of demand with VOT v, for any $v \in V$. The p_v values are unknown to a toll operator. For simulation purposes, we choose the VOT distribution $(p_v | v \in V)$ and σ_d to be identical for all origin–destination pairs. Though dynamic traffic assignment models have been used in the literature for optimization of toll prices for MLs (Zhang et al., 2018), we focus on real-time optimization of toll prices and ignore route-choice equilibration of travelers (A#3). The lane choice models are discussed in Section 2.2.

Traffic flow models can either be microscopic or macroscopic. With the exception of Belletti et al. (2017), all other Deep-RL models in transportation domain use microsimulation to capture the vehicle-to-vehicle interactions. In this article, we use macroscopic models to represent traffic flow for the simplicity they provide. In contrast to the cell-based representation of managed lane network in macroscopic traffic models from the literature, where MLs and GPLs are modeled as part of the same cell (Tan and Gao, 2018; Yang et al., 2012; Dorogush and Kurzhanskiy, 2015), we divide each link into individual cells, where the links for GPLs are separate from that of MLs. This choice lets us use the cell transmission model (CTM) equations from Daganzo (1995) for modeling traffic flow. Let $\mathscr{C}_{(i,j)}$ represent the set of all cells for link $(i, j) \in A$ and $\mathscr{C} = \bigcup_{(i,j)\in A} \mathscr{C}_{(i,j)}$ denote the set of all cells in the network. The length of each cell $c \in \mathscr{C}$, denoted by l_c, is determined as usual (the distance traveled at free flow in time Δt) (Daganzo, 1995), and is assumed constant for all links in the network (A#4). We thus require all link lengths to be integral multiples of the cell length. Let l_{ij}, v_{ij}, $q_{\max,ij}$, w_{ij}, and $k_{\text{jam},ij}$ represent the length, free-flow speed, capacity, back-wave speed, and jam density, respectively, for link $(i, j) \in A$ as its fundamental diagram parameters, which we assume has a trapezoidal shape (A#5).

A toll operator is assumed to manage the toll rate at each on-ramp and diverge point beyond a diverge on a ML (A#6). We assume this toll structure in contrast to the generic structure of separate toll values for each origin–destination (OD) pair, like in Yang et al. (2012) and Tan and Gao (2018), because it inherently models the constraint that traveling longer distance on the ML levies a higher toll than traveling shorter distance. For a detailed discussion on various options to charge toll on a managed lane network with multiple accesses, see Pandey and Boyles (2019). Let A_{toll} represent the links where tolls are collected. Fig. 1 highlights these links in bold. We denote the toll charged on link $(i, j) \in A_{\text{toll}}$ for any $t \in \mathscr{T}$ by $\beta_{ij}(t)$.

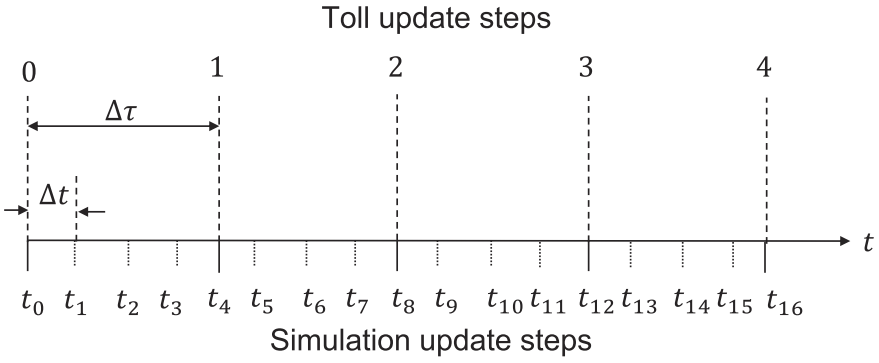


Fig. 2. Representation of a time scale where tolls update after $\Delta\tau$ units and the simulation updates after Δt units.

Table 1
Categorization of lane choice models for managed lanes with multiple entrances and exits.

Number of VOT classes	Number of routes over which the utility(s) is (are) compared	Deterministic or Stochastic	Reference(s) in the literature using this lane choice
Single	Two	Deterministic	Gardner et al. (2013)
Single	Two	Stochastic	Tan and Gao (2018), Toledo et al. (2015), Zhu and Ukkusuri (2015), Yang et al. (2012)
Single	Decision routes	Deterministic	None
Single	Decision routes	Stochastic	None
Multiple	Two	Deterministic	Gardner et al. (2015)
Multiple	Two	Stochastic	None
Multiple	Decision routes	Deterministic	Pandey and Boyles (2018a,b)
Multiple	Decision routes	Stochastic	None

2.2. Lane choice model

Travelers make lane choice decisions at each diverge location (nodes a, c, f, and h in Fig. 1), where information about the current travel time and toll values is displayed. We assume that the information about the current travel time is provided by measuring instantaneous travel time with no time lag (A#7), and that all travelers make their lane choice decision only using the instantaneous/real-time information (A#8). Assumptions A#7 and A#8 are only made for simulation purposes, as the Deep-RL model only requires the realization of lane choices in form of observed loop detector measurements. If we have an estimate of experienced travel time on each route, the simulations can be based on experienced travel time. Assumptions A#3 and A#8 are related: because we assume no prior experience for the drivers, users do not find an equilibrium over route choices. Considering dynamic route-choice equilibrium while optimizing a dynamic stochastic control is a complex problem and will be studied as part of the future work (FW#1).

Conceptually, the lane choice models can be categorized based on three characteristics: the number of routes over which travelers compare the utility, whether or not the lane choice is stochastic/deterministic, and the heterogeneity in vehicles' value of time (single class vs multiple classes). Commonly used binary Logit model assumes stochastic lane choice over two routes connecting current diverge to the destination, while the decision route model evaluates deterministic lane choice of multiple vehicle classes comparing utilities over a set of routes connecting current diverge to the merge after the first exit from the ML (Pandey and Boyles, 2018a). A recent analysis in Pandey and Boyles (2019) showed that the decision route model has least error compared to the optimal route choice model for rational travelers and offers an efficient way for simulating route choices over a corridor, thus providing a potential for speeding up Deep-RL training.

Table 1 shows the combinations of categories and models used in the literature. Certain combination have not been used directly, but they could be used. For example, combining decision routes with stochastic lane choice can result in models like multinomial logit or mixed logit, but the assumption that the utilities across overlapping routes are independent may not hold true.

The Deep-RL algorithm developed in this article is agnostic to the lane choice model. For simulation purposes, we focus our attention on two models: multiple VOT classes with two routes and stochastic choice (*multiclass binary logit model*) and multiple VOT classes with decision routes and deterministic choice (*multiclass decision route model*). For simulation purposes, we evaluate the utility of a route as the linear combination of the toll and route's travel time, converted to the same units using the VOT for the class (A#9).

2.3. Partially observable markov decision process

MDPs are a discrete time stochastic control process that provide a framework for solving problems that involve sequential decision making (Sutton and Barto, 2018). At each time step, the system is in some *state*. The decision maker takes an *action* in that *state*, and the system transitions to the next *state* depending on the transition probabilities, which are only a function of the current *state* and the *action* taken (called the Markov property). Given an *action*, this transition from one *state* to the other generates a reward for each time step and the decision maker seeks to maximize the expected reward across all time steps. Control problems in transportation do not necessarily have the Markov property because of the temporal dependence of congestion pattern. However, by including the simulation time as part of the state, they can be formulated as an MDP.

Partially observable Markov decision processes (POMDPs) are MDPs where the state at any time step is not known with certainty, that is, the state is not fully observable. For the dynamic pricing problem where a toll operator does not have access to traffic information throughout the network but only at certain locations, POMDPs are a suitable choice. We define the control problem for determining the optimal toll as an POMDP with following components:

- **Timestep**: Tolls are to be optimized over a finite time horizon for each time $k \in \mathscr{T}_\tau$. A finite horizon can represent a morning or an evening peak period on a corridor, or an entire day.
- **State**: We first define $x_c^z(t)$ as the number of vehicles in cell $c \in \mathscr{C}$ belonging to class $z \in Z$ at time $t \in \mathscr{T}$, where $Z = \{(v, d) | v \in V, d \in N_d\}$ is the set of all classes, disaggregated by the VOT value and the destination of the vehicle (the origin of a vehicle does not influence lane choice once the vehicle is on the road and is thus ignored). For ML networks where high occupancy vehicles pay a different toll than single/low occupancy vehicles, we can extend Z to include the occupancy level of

vehicles, but we leave that analysis for future work (FW#2). The dimensionality of Z impacts the computational performance of the multiclass cell transmission model. Similar to the non-atomic flow assumption commonly used in the transportation literature, we consider $x_c^z(t)$ to be a non-negative real number. We denote the state of the POMDP by s comprising of the current toll update step $k \in \mathscr{T}_\tau$ and the values $x_c^z(t_{k\Delta\tau})$ for all cells $c \in \mathscr{C}$ and class $z \in Z$. Thus, the state space S can be written as Eq. (2.1). Allowing $\Delta\tau$ to be greater than Δt ($m > 1$) reduces the size of state space compared to choosing $m = 1$, which improves the computational efficiency.

$$S = \{(k, x_c^z(t_{k\Delta\tau}))|k \in \mathscr{T}_\tau, c \in \mathscr{C}, z \in Z\} \tag{2.1}$$

- **Observation**: In our model, the observation is done using loop detectors. The detectors measure the total number of vehicles going from one cell to the next and cannot distinguish between vehicles belonging to different classes, so the state is not fully observable. The observation space depends on the location of detectors. We conduct sensitivity analyses with respect to changes in the observation space later in the text. Let $\boldsymbol{o}(s)$ denote the observation vector for state s and comprise of the measurement of total number of vehicles on each link $(i, j) \in A_{\text{loop}} \subseteq A$ which has a loop detector installed at beginning and end.[1] That is, $\boldsymbol{o}(s) = \left\{ \sum_{z\in Z} \sum_{c\in\mathscr{C}_{(i,j)}} x_c^z(t_{k\Delta\tau}) \middle| \left(i, j\right) \in A_{\text{loop}} \right\}$. We assume that we can learn the total number of vehicles on any link by tracking the number of vehicles entering the link (measured at an upstream detector) and the number of vehicles leaving the link (measured at a downstream detector) (A#10). The actual observation is assumed to be Gaussian random variable with the mean as specified and the standard deviation σ_o which models the noise in loop detector measurements. We project negative values of observation, if any, to zero.
- **Action**: Action a in state s is the toll $\beta_{ij}(t_{k\Delta\tau})$ charged for a toll link $(i, j) \in A_{\text{toll}}$, where $\beta_{ij}(\cdot) \in [\beta_{\min}, \beta_{\max}]$. The action is modeled as a continuous variable; the values can be rounded to nearest tenth of a cent or dollar if desired.
- **Transition function**: Because we use model-free RL methods for solving the POMDP, the learning algorithm is agnostic to the choice of transition function. In our experiments, we simulate the transition from a state to a new state, given an action, using the traffic flow equations from the CTM model which incorporates the lane choice behavior of travelers. For simulation purposes, we assume that traffic flow throughout the network is deterministic except at diverges where the lane choices of travelers may be stochastic (A#11). We use a multiclass version of the CTM model similar to the model in Pandey and Boyles (2018a).
- **Reward**: The reward obtained after taking action a in state s, denoted by $r(s, a)$, depends on the choice of tolling objective. We consider two objectives, revenue maximization and total system travel time (TSTT) minimization, with following definitions of reward:
  - Revenue maximization:

$$r^{\text{RevMax}}\left(s, a\right) = \sum_{x=k\Delta\tau}^{(k+1)\Delta\tau-1} \sum_{(i,j)\in A_{\text{toll}}} \left(\beta_{ij}(t_{k\Delta\tau}) \sum_{(h,i)\in A} y_{hij}(t_x) \right), \tag{2.2}$$

    where $y_{hij}(t)$ is the total flow moving from link $(h, i) \in A$ to $(i, j) \in A$ from time step t to time step $t + \Delta t$
  - Total system travel time minimization:

$$r^{\text{TSTTMin}}\left(s, a\right) = -\left(\sum_{x=k\Delta\tau}^{(k+1)\Delta\tau-1} \sum_{c\in\mathscr{C}} \sum_{z\in Z} x_c^z(t_x) \right), \tag{2.3}$$

    where the negative sign ensures that reward maximization is equivalent to TSTT minimization.

For the dynamic pricing problem, revenue-maximizing tolls often have a JAH nature where the GPLs are jammed to congestion earlier in the simulation to attract more travelers towards the ML later in the simulation generating more revenue (Göçmen et al., 2015; Pandey and Boyles, 2018a). This undesirable characteristic of optimal policy is also seen in other applications of RL. For example, for ATSC a simpler definition of reward that maximizes amount of flow during a cycle may lead to "evil" optimal policies, where the controller agent holds congestion on the mainline and then gains a larger reward by extending the greens for the main approach (Shabestary and Abdulhai, 2018). Similarly, Van der Pol and Oliehoek (2016) show that with inappropriate definitions of reward, the signal control policy may have unusual flips from green to red.

To overcome the undesired JAH nature, we use reward shaping methods that modify the reward definitions such that the optimal policies have less or no JAH behavior (discussed later in Section 4.4). For reward shaping, we quantify the JAH behavior using two statistics defined as a numeric value at the end of simulation. The first statistic, JAH_1, measures the maximum of difference between the number of vehicles in GPLs to the number of vehicles in MLs across all time steps. It is defined as in Eq. (2.4), where $A_{\text{GPL}}(A_{\text{ML}})$ are links on the GPL (ML).

[1] For Fig. 1, $A_{\text{loop}} = \{(o, a), (a, c), (c, e), (d, f), (g, h), (h, j)\}$.

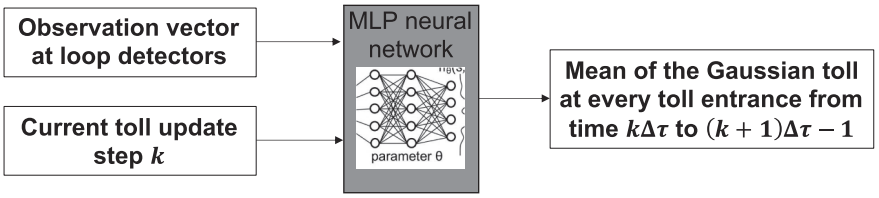


Fig. 3. Abstract representation of the policy including toll update step as an input.

$$\text{JAH}_1 = \max_{t\in\mathscr{T}} \left(\sum_{(i,j)\in A_{\text{GPL}}} \sum_{c\in\mathscr{C}_{(i,j)}} \sum_{z\in Z} x_c^z(t) - \sum_{(i,j)\in A_{\text{ML}}} \sum_{c\in\mathscr{C}_{(i,j)}} \sum_{z\in Z} x_c^z(t) \right) \tag{2.4}$$

The value of JAH_1 is dependent on network properties like number of lanes in GPLs and MLs. We also define an alternate statistic JAH_2 that is network independent. We first define $\zeta(t)$, as in Eq. (2.5), as the difference between the ratio of current number of vehicles to the maximum number of vehicles allowed in each cell (corresponding to jam density) for all cells on GPLs with that of MLs.

$$\zeta(t) = \frac{\sum_{(i,j)\in A_{\text{GPL}}} \sum_{c\in\mathscr{C}_{(i,j)}} \sum_{z\in Z} x_c^z(t)}{\sum_{(i,j)\in A_{\text{GPL}}} \sum_{i\in\mathscr{C}_{(i,j)}} l_{ij} k_{\text{jam},ij}} - \frac{\sum_{(i,j)\in A_{\text{ML}}} \sum_{i\in\mathscr{C}_{(i,j)}} \sum_{z\in Z} x_i^z(t)}{\sum_{(i,j)\in A_{\text{ML}}} \sum_{i\in\mathscr{C}_{(i,j)}} l_{ij} k_{\text{jam},ij}} \tag{2.5}$$

JAH_2 can then be defined as a maximum value of $\zeta(t)$ across all time steps, as in Eq. (2.6). The value of JAH_2 varies between $[-1, 1]$ with a high positive value indicating more congestion on GPLs before congestion set in the ML.

$$\text{JAH}_2 = \max_{t\in\mathscr{T}} \zeta(t) \tag{2.6}$$

For the given POMDP, a policy $\pi_\theta(a|\boldsymbol{o}(s))$ denotes the probability of taking action a given observation $\boldsymbol{o}(s)$ in state s. We consider stochastic policies parameterized by a vector of real parameters θ. For example, for a policy replaced by a neural network, θ represents the flattened weights and biases for the nodes in the network. Since the action space for the POMDP is continuous, the neural network outputs the mean of the Gaussian distribution of tolls which is then used to sample continuous actions. For simplicity in Deep-RL training, we assume the covariance of the joint distribution of actions to be a diagonal matrix with constant diagonal terms (A#12). Fig. 3 shows a schematic of the parameterized representation of the policy which takes in the input of observations across the network and returns the mean of the Gaussian toll values for all toll links. MLP stands for multi-layer perceptron which is a feed-forward neural network architecture.

2.4. Episodic reinforcement learning

In an episodic reinforcement learning problem, an agent's experience is broken into episodes, where an episode is a sequence with a finite number of states, actions, and rewards. Since the POMDP introduced in the previous subsection is finite-horizon, the simulation terminates at time $T/\Delta t$. Thus, an episode is formed by a sequence of states, actions, and rewards for each time step $k \in \mathscr{T}_\tau$.

We first define a trajectory $\aleph$ as a sequence of states and actions visited in an episode, that is $\aleph = (s_0, a_0, s_1, a_1, \cdots, s_{|\mathscr{T}_\tau|-1})$, where s_k is same as the state defined earlier indexed by the time k in that state. Let $r(s_k, a_k)$ be denoted by r_k for all $k \in \mathscr{T}_\tau$.

The goal of the RL problem is to find a policy that maximizes the expected reward over the entire episode. The optimization problem can then be written as following:

$$\max_{\pi_\theta(\cdot)} J(\pi_\theta) = \mathbb{E}_\aleph \left[R(\aleph) \middle| \pi \right] \tag{2.7}$$

$$R(\aleph) = \sum_{k\in\mathscr{T}_\tau} r_k, \tag{2.8}$$

where, $\mathbb{E}_\aleph[R(\aleph)|\pi] = \int R(\aleph) p_\pi(\aleph) d\aleph$ is the expected reward over all possible trajectories obtained after executing policy π with $p_\pi(\aleph)$ as the probability distribution of trajectories obtained by executing policy π.[2] We do not discount future rewards because tolls are optimized over a short time period (like a day or a morning/evening peak).

We define a few additional terms used later in the text. Let $V^\pi(s_k) = \mathbb{E}_\aleph \sum_{k'=k}^{|\mathscr{T}_\tau|} r_{k'}$ be the value function which evaluates the expected reward obtained from state s_k till the end of episode following policy π. Similarly, we define the Q-function, denoted by $Q^\pi(s_k, a_k)$, as the expected reward obtained till the end of episode from state s_k after taking action a_k and following policy π thereafter. Last, the advantage function $A^\pi(s_k, a_k) = Q^\pi(s_k, a_k) - V^\pi(s_k)$, defined as the difference between Q-function and value function, determines how much better or worse is an action than other actions on average, given the current policy.

[2] Defining an expectation conditioned over a function (π) instead of a random variable is a slight abuse of notation, but is commonly used in the RL literature.

The solution of this POMDP is a vector θ^* that determines the policy which optimizes the objective under certain constraints on the policy space. Commonly considered policy constraints for the dynamic pricing of MLs include the following:

1. Tolls levied for a longer distance are higher than tolls levied for a shorter distance from the same entrance: with the choice of tolling structure (assumption A#6) where tolls are charged at every diverge, this constraint is already satisfied.
2. The ML is always operated at a speed higher than the minimum speed limit (called the speed-limit constraint): in our model, we allow violation of this constraint on the ML. We observe that, given the stochasticity in lane choice of travelers and demand, bottlenecks can occur at merges and diverges which can result in an inevitable spillover on managed lanes during congested cases. Thus, a hard constraint keeping the ML congestion free throughout the learning period is not useful. We instead quantify the violation of the speed-limit constraint using the time–space diagram of the cells on the ML. We define `%-violation` as the proportion of cell-timestep pairs on the time–space diagram where the speed limit constraint is violated, expressed as percentage. Mathematically,

$$\%-\text{violation} = \frac{\sum_{(i,j)\in A_{\text{ML}}} \sum_{c\in\mathscr{C}_{(i,j)}} \sum_{t\in\mathscr{T}} I_c^t}{\left|\mathscr{T}\right| \sum_{(i,j)\in A_{\text{ML}}} \left|\mathscr{C}_{(i,j)}\right|} \times 100, \tag{2.9}$$

   where I_c^t is an indicator variable which is 1 if the number of vehicles in the cell c in time step t is higher than the desired number of vehicles in the cell and 0 otherwise. The desired number of vehicles in each cell is determined from the density corresponding to the minimum speed limit on the fundamental diagram. As discussed in Section 4, allowing the speed-limit constraint to be violated in our model is not restrictive as the best-found policies for each objective have `%-violation` values of less than 2% for all networks tested.
3. Toll variation from one time step to the next is restricted: we do not explicitly model this constraint. If the tolling horizon is "sufficiently" large (say 5 min), a large change in tolls from one toll update to the next can be less of a problem. In our experiments, the optimal tolls are structured and do not oscillate significantly.
4. Tolls are upper and lower bounded by a value: we model this by clipping the toll output by the function approximator within the desired range $[\beta_{\min}, \beta_{\max}]$.

Next, we discuss the solution methods used to solve the POMDP using Deep-RL methods and other heuristics.

3. Solution methods

3.1. Deep reinforcement learning algorithms

We start with a brief review of reinforcement learning algorithms. There are two broad categories: model-based RL algorithms where transition probabilities and reward functions are accessible to the agent (for example, Imagination-augmented Agents (I2A) algorithm (Racanière et al., 2017)), and model-free RL algorithms where the agent learns actions with better reward through direct interactions with the environment. Model-based methods work well because they allow the agent to strategically plan ahead considering the future rewards and transitions. However, for the toll pricing problem, with high dimensional state space and continuous action space, planning over all possible future states and actions is difficult and thus we prefer using model-free RL methods.

The model-free RL algorithms are further subdivided based on the characteristic of the MDP that is learned from interactions with the environment. These include value-based methods and policy-based methods. The value-based methods try to learn the value functions and use approaches based on dynamic programming to solve the problem, while the policy-based methods try to learn the policy directly based on the observations. Deep Q-learning (DQN) (Mnih et al., 2013) is one of the most popular value-based method where function approximators for value functions are replaced by neural networks. Policy-based methods work well with continuous state and action spaces, making it a preferred choice for the toll optimization problem. These include algorithms like policy gradient, advantage actor critic (A2C), and proximal policy optimization (PPO) (Schulman et al., 2017). Derivative-free optimization and gradient-based optimization are two subcategories among policy-based methods. We focus on the methods relying on derivatives as they are considered to be data efficient (Schulman, 2016).

Deep-RL algorithms are also subdivided into on-policy and off-policy algorithms based on how agent's experiences are used to learn better policies. On-policy algorithms like A2C and PPO learn the value of the policy as the agent carries out actions and use those learned values directly for control and for improving the policy. In contrast, off-policy algorithms improve the policy using data that was collected across different past interactions, making these algorithms more sample efficient. DQN with Experience Replay is one of the most popular off-policy algorithm. Recently, off-policy algorithms incorporating the advantages of policy optimization have been proposed for environments with continuous control. These include deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015) and soft actor-critic (SAC) (Haarnoja et al., 2018), to name a few. While the algorithms for deep RL are continuously evolving, DDPG and SAC are considered state-of-the-art and outperform PPO, A2C, and DQN on several continuous control tasks in finding policies with better reward in fewer environment interactions.

Fig. 4 shows the taxonomy of the RL models in the literature. Providing details on the state-of-the-art algorithms to solve RL problems is out of the scope of this work; we refer the reader to references within Schulman (2016) and SpinningUp documentation (OpenAI, 2019). In this article, we choose two of the commonly used policy gradient algorithms for solving the continuous control

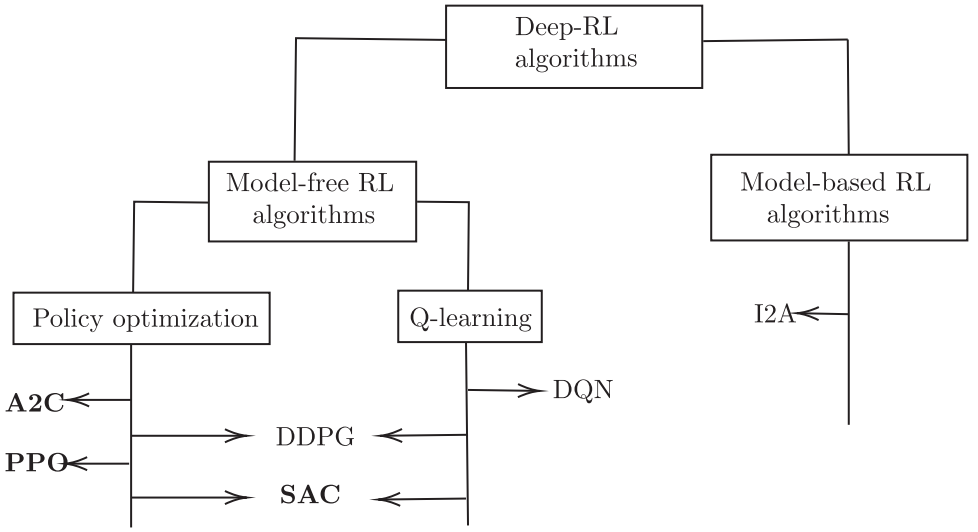


Fig. 4. Partial taxonomy of RL algorithms in the literature. The algorithms used in this study are highlighted in bold. (Adapted from SpinningUp documentation OpenAI (2019)).

problem: A2C and PPO, and the off-policy maximum entropy Deep-RL algorithm, SAC. We describe the necessary details of the algorithms next.

The A2C and PPO algorithms use the derivative of the objective function with respect to the policy parameters to improve them using stochastic gradient descent. The methods differ in calculation of the derivatives and the update of parameter θ. We can express the derivative of $J(\pi_\theta)$ with respect to θ as:

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\aleph}[R(\aleph)|\pi] \tag{3.1a}$$

$$= \nabla_\theta \int_{\aleph} P\left(\aleph \middle| \theta\right) R(\aleph) d\aleph \tag{3.1b}$$

$$= \int_{\aleph} \nabla_\theta P\left(\aleph \middle| \theta\right) R(\aleph) d\aleph \tag{3.1c}$$

$$= \int_{\aleph} P\left(\aleph \middle| \theta\right) \nabla_\theta \log P\left(\aleph \middle| \theta\right) R(\aleph) d\aleph \left(\text{since} \nabla_\theta \log P\left(\aleph \middle| \theta\right) = \frac{1}{P(\aleph|\theta)} \nabla_\theta P\left(\aleph \middle| \theta\right)\right) \tag{3.1d}$$

$$= \mathbb{E}_{\aleph}[\nabla_\theta \log P(\aleph|\theta) R(\aleph)] \tag{3.1e}$$

$$= \mathbb{E}_{\aleph}\left[\sum_{k=0}^{|\mathscr{T}_\tau|} \nabla_\theta \log\left(\pi_\theta\left(a_k \middle| s_k\right)\right) R(\aleph)\right], \tag{3.1f}$$

where we first convert the probability of a trajectory into a product of the probabilities of taking certain actions in each state, and then convert this product into a sum. As a result, the derivative in the RHS of Eq. (3.1f) can be easily obtained by performing back propagation on the policy neural network.

The expectation in Eq. (3.1f) can be approximated by averaging over a finite number of trajectories. Let $\mathcal{N} = \{\aleph_i | i \in 1, 2, ...\}$ be the set of trajectories obtained using policy $\pi_\theta(\cdot)$. Then, we can write:

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{|\mathcal{N}|} \sum_{\aleph \in \mathcal{N}} \left[\sum_{k=0}^{|\mathscr{T}_\tau|} \nabla_\theta \log\left(\pi_\theta\left(a_k \middle| s_k\right)\right) R(\aleph)\right]. \tag{3.2}$$

In the above formulation the likelihood of actions taken along the trajectory is affected by reward over entire trajectory. However, it is more intuitive for an action to influence the reward obtained only after the time step when it was implemented. It can be shown that the right hand side of the expression in Eq. (3.2) is equivalent to the following expression:

$$\frac{1}{|\mathcal{N}|} \sum_{\aleph \in \mathcal{N}} \left[\sum_{k=0}^{|\mathscr{T}_\tau|} \nabla_\theta \log\left(\pi_\theta\left(a_k \middle| s_k\right)\right) \widehat{R}(k)\right], \tag{3.3}$$

where $\widehat{R}(k)$ is the reward-to-go function at time k, given by $\widehat{R}(k) = \sum_{k'=k}^{|\mathscr{T}_\tau|} r_{k'}$. This new expression for the gradient of the objective

requires sampling of fewer trajectories and generates a low-variance sample estimate of the gradient.

Additionally, the variance can be further reduced by using the advantage function estimates instead of reward-to-go function (Schulman et al., 2015). A2C uses the following form for approximating the derivative:

$$\nabla_\theta J\left(\pi(\theta)\right) \approx \frac{1}{|\mathcal{N}|}\sum_{\aleph\in\mathcal{N}}\left[\sum_{k=0}^{|\mathscr{T}_\tau|}\nabla_\theta \log\left(\pi_\theta\left(a_k\middle|s_k\right)\right)\widehat{A}_k\right], \tag{3.4}$$

where $\widehat{A}_k$ is the estimate of advantage function, $A^{\pi_\theta}(s_k, a_k)$, from current time k till the end of episode, following the policy from which the given trajectory is sampled. We use the generalized advantage estimation (GAE) technique to estimate $\widehat{A}_k$ which requires an estimate of the value function (Schulman et al., 2015). We use value function approximation to estimate of $V^\pi(s_k)$ using a neural network as the functional approximator. Let $\widehat{V}_\phi(s_k)$ denote the estimate of $V^\pi(s_k)$, parameterized by a real vector of parameters ϕ. The algorithm starts with an estimate of ϕ (ϕ_0) and iteratively improves it by minimizing the squared difference with the reward-to-go value from the trajectory. The update in ϕ parameters are evaluated using Eq. (3.5):

$$\phi_{n+1} = \text{argmin}_\phi \frac{1}{|\mathcal{N}||\mathscr{T}_\tau|}\sum_{\aleph\in\mathcal{N}}\sum_{k=0}^{|\mathscr{T}_\tau|}\left(V_\phi(s_k) - \widehat{R}(k)\right)^2. \tag{3.5}$$

More details on GAE are provided in Schulman et al. (2015).

A2C updates the value of θ parameter from iteration n to $n+1$ using the standard gradient ascent formula:

$$\theta_{n+1} = \theta_n + \alpha\nabla_\theta J(\pi(\theta_n)). \tag{3.6}$$

In Eq. (3.6), an inappropriate choice of the learning rate α can lead to large policy updates from one iteration to the next which can cause the objective values to fluctuate. The PPO algorithm modifies the policy update to take the biggest possible improvement using the data generated from current policy while ensuring improvement in the objective. It performs specialized clipping to discourage large changes in the policy. The policy update for PPO is given by:

$$\theta_{n+1} = \text{argmax}_\theta \frac{1}{|\mathcal{N}|}\sum_{\aleph\in\mathcal{N}}\left[\sum_{k=0}^{|\mathscr{T}_\tau|}\min\left(r_k(\theta)\widehat{A}^{\pi_{\theta_n}}(s_k, a_k), \text{clip}(r_k(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}^{\pi_{\theta_n}}(s_k, a_k)\right)\right], \tag{3.7}$$

where $r_k(\theta)$ is the ratio of probabilities following a policy and the policy in the current iteration (θ_n) given by Eq. (3.8), and the `clip` (·) function, given by Eq. (3.9), restricts the value of first argument between the next two arguments.

$$r_k(\theta) = \frac{\pi_\theta(a_k|s_k)}{\pi_{\theta_n}(a_k|s_k)} \tag{3.8}$$

$$\text{clip}\left(r, 1-\epsilon, 1+\epsilon\right) = \begin{cases} 1-\epsilon, & \text{if} r \leqslant 1-\epsilon \\ r, & \text{if} 1-\epsilon < r < 1+\epsilon \\ 1+\epsilon, & \text{if} r \geqslant 1+\epsilon. \end{cases} \tag{3.9}$$

The clipping operation selects the policy parameters in the next iteration such that the ratio of action probabilities in iteration $n+1$ to iteration n are between $[1-\epsilon, 1+\epsilon]$, where ϵ is a small parameter, typically 0.01. Policy updates for PPO can be solved using the Adam gradient ascent algorithm, a variant of stochastic gradient ascent with adaptive learning rates for different parameters (Kingma and Ba, 2014; Schulman et al., 2017). The pseudo-code for both algorithms is presented in Algorithm 1.

Algorithm 1. A2C and PPO algorithms for dynamic pricing OpenAI (2019)

Input: initialize policy parameters θ_0 and value function parameters ϕ_0
for $n = 0, 1, 2, \cdots$ **do**
  Collect set of trajectories $\mathcal{N}_n = \{\aleph_n\}$ by running policy $\pi_n = \pi_{\theta_n}$ in the environment
  Compute rewards to go $\widehat{R}_k$
  Compute advantage estimates using rewards-to-go and generalized advantage estimation
  Update policy parameters using either A2C or PPO update:
  • **A2C**: Estimate policy gradients using Eq. (3.4) and update policy parameters using Eq. (3.6), or
  • **PPO**: Update policy parameters by solving Eq. (3.7) using Adam gradient ascent algorithm
  Update value function approximation parameter (used for advantage estimation) in Eq. (3.5) using Adam gradient descent
end for

The soft actor-critic algorithm trains the policy in an off-policy fashion by learning the optimal Q-function associated with state-action pair combination. A key feature of SAC is entropy regularization which adds an entropy term to the optimization objective resulting in more exploration and allowing it to escape bad local optimum. SAC concurrently learns a policy and two Q-functions each approximated using different neural network parameters. The Q-function parameters are learned by minimizing the least square Bellman error against a single shared target. This shared target is initialized to the same value as the Q-function parameters and is iteratively updated using Polyak-Ruppert averaging. SAC considers the minimum among the two Q-functions to determine an update to the policy resulting in stable learning profile. Explaining the details of the SAC algorithm is beyond the scope of the text. We refer

the reader to Haarnoja et al. (2018) for additional details. We note that while SAC explores using a stochastic policy, the output policy at the end of simulation is deterministic with tolls set as the output of neural network.

For the experiments, we develop a new RL environment for macroscopic simulation of traffic similar to the current RL benchmarks (called "gym" environments) and customize the open-source implementation of the three algorithms provided by OpenAI Spinningup (OpenAI, 2019) to work with our new environment. Appendix A shows the hyperparameters used for training A2C, PPO, and SAC algorithms.

3.2. Feedback control heuristic

We compare the performance of Deep-RL algorithms against a feedback control heuristic based on the measurement of total number of vehicles in the links on ML. We use the proportional-integral-derivative (PID) controller commonly used to regulate engineering systems. The feedback-control heuristics in Pandey and Boyles (2018a) and Tan and Gao (2018) are a special case of PID controller. Furthermore, versions of PID controller are commonly used in current industry applications. To formally state the PID controller, we first define a few terms.

Define $\mathrm{ML}(i, j)$ as the set of links on the ML used by a traveler upon first entering the ML using the toll link $(i, j) \in A_{\text{toll}}$ until the next merge or diverge. For the network in Fig. 1, $\mathrm{ML}(a, b) = \{(b, d)\}$, $\mathrm{ML}(c, d) = \{(d, f)\}$, $\mathrm{ML}(f, i) = \{(f, i)\}$, and $\mathrm{ML}(h, i) = \{(i, k)\}$. This definition allows the sets $\mathrm{ML}(i, j)$ to be mutually exclusive and exhaustive in the space of all links on the ML. That is,

$$\mathrm{ML}(i, j) \cap \mathrm{ML}(k, l) = \Phi \forall\ (i, j) \in A_{\text{toll}}, (k, l) \in A_{\text{toll}}, (i, j) \neq (k, l)$$

$$\bigcup_{(i,j)\in A_{\text{toll}}} \mathrm{ML}\Big(i, j\Big) = A_{\mathrm{ML}}.$$

We assume that the PID controller updates the tolls for each toll link $(i, j) \in A_{\text{toll}}$ based on the density observations on links in $\mathrm{ML}(i, j)$, that is, detectors are installed on each link in the ML and only those detectors are used to update the toll (A#13).

The toll value for an update time $(k + 1) \in \mathscr{T}_\tau$ for link $(i, j) \in A_{\text{toll}}$ is based on the toll value in the previous update step (k) adjusted by the difference between the desired and current numbers of vehicles on the links in the set $\mathrm{ML}(i, j)$. This difference is defined as the error in the PID heuristic and is represented by $e_{ij}(k)$. Mathematically,

$$e_{ij}(k) = (X_{\mathrm{ML}(i,j)}(k) - X^{\text{desired}}_{\mathrm{ML}(i,j)}), \tag{3.10}$$

for all $(i, j) \in A_{\text{toll}}$, where $X_{\mathrm{ML}(i,j)}(k)$ is the total number of vehicles on links in $\mathrm{ML}(i, j)$ before updating tolls at time $k + 1$ and $X^{\text{desired}}_{\mathrm{ML}(i,j)}$ be the desired value of the number of vehicles on the links in $\mathrm{ML}(i, j)$. A typical desired value is the number of vehicles corresponding to the critical density on the ML link. We generalize the desired number of vehicles by defining $X^{\text{desired}}_{\mathrm{ML}(i,j)}$ as:

$$X^{\text{desired}}_{\mathrm{ML}(i,j)} = \sum_{(g,h)\in \mathrm{ML}(i,j)} \eta k_{\text{critical},(g,h)} l_{gh}, \tag{3.11}$$

where, $k_{\text{critical},(g,h)}$ is the critical density for link $(g, h) \in A$ and η is the scaling parameter varying between $(0, 1]$ that sets the desired number of vehicles to a proportion value of the number of vehicles at critical density.

Next, the toll update is specified based on the error terms. The PID-controller updates tolls which is proportional to three terms: the error, its integral since the start of simulation, and its derivative at the current time step. We compute the numerical integral using trapezoidal Reimann sum and the numerical derivative using backward difference method. The toll update is given by Eq. (3.12),

$$\beta_{ij}(t_{(k+1)\Delta\tau}) = \beta_{ij}(t_{k\Delta\tau}) + K_p e_{ij}(k) + K_i \int_0^k e_{ij}(s)ds + K_d \frac{de_{ij}(k)}{dk}, \tag{3.12}$$

where, K_p, K_i, and K_d are the proportional, integral, and derivative regulator parameters. These parameters control the influence of difference between the desired and current number of vehicles on the toll update. For more details on the PID controller, refer Åström and Murray (Åström and Murray, 2010).

We calibrate the PID controller for different values of parameters η, K_p, K_i, and K_d. In principle, all parameters can vary with time and the toll location; however, determining the "optimal" variability in these parameters is a control problem in itself, exploring which is left as part of the future work (FW#3).

In Section 4.5 we also compare the performance of algorithms making the *full observability* assumption against the Deep-RL algorithms which do not make that assumption. We choose two algorithms from our previous work: the algorithm based on value function approximation (VFA) using look-up tables (Pandey and Boyles, 2018a), and the multiagent reinforcement learning algorithm that learns value functions separately for each toll gantry (SparseV algorithm) (Pandey and Boyles, 2018b). Comparing the performance of Deep-RL methods against the hybrid MPC method in Tan and Gao (2018) requires extensive analysis and will be a part of the future work (FW#4).

4. Experimental analysis

4.1. Preliminaries

We conduct our analysis on four different networks. The first is a network with single entrance and single exit (SESE) commonly

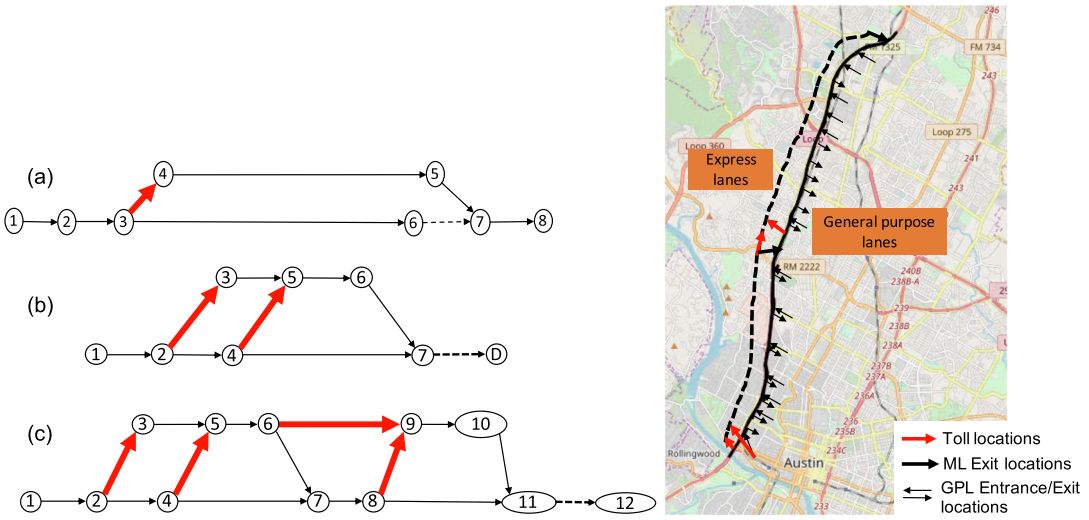


Fig. 5. Abstract representation of (a) single entrance single exit (SESE) network, (b) double entrance single exit (DESE) network, (c) LBJ network, and (d) Northbound MoPac express lane network (latitude-longitude locations of MLs are shifted to the left to show the locations of toll points and exits from the managed lane). The tolls are collected on the links with higher thickness.

used in the managed lane pricing literature. The next two are the double entrance single exit (DESE) network and the network for toll segment 2 of the LBJ TEXpress lanes in Dallas, TX (LBJ). The DESE network includes two toll locations for modeling *en route* lane changes. The LBJ network has four toll locations. Last is the network of the northbound Loop 1 (MoPac) Express lanes in Austin, TX. The MoPac network has three entry locations to the MLs and two exit locations.

Fig. 5 shows the networks, where the thick lines denote the links where tolls are collected. The demand distribution for the first three networks is artificially generated and follows a two-peak pattern (refer to the original demand curve in Fig. 6a), while the demand for the MoPac network is derived from a dynamic traffic assignment model of the Travis County region. There are a total of 105 origin–destination pairs in the MoPac network with a total demand of 49, 273 vehicles using the network in three hours of the evening peak.

Table 2 shows the values of parameters used for different networks. Five VOT classes were selected for each network and the same VOT distribution was used. Fig. 6b shows this VOT distribution (labelled "original"; in some experiments we vary this distribution.).

A feedforward multilayer perceptron was selected as the neural network. The values of hyperparameters for Deep-RL training obtained after tuning are included in Appendix A. Each network was simulated for up to 1200 episodes or until the objective values converge.

4.2. Validating JAH statistics

In this subsection, we discuss how the JAH statistics defined in Eqs. (2.4) and (2.6) are meaningful in capturing the jam-and-

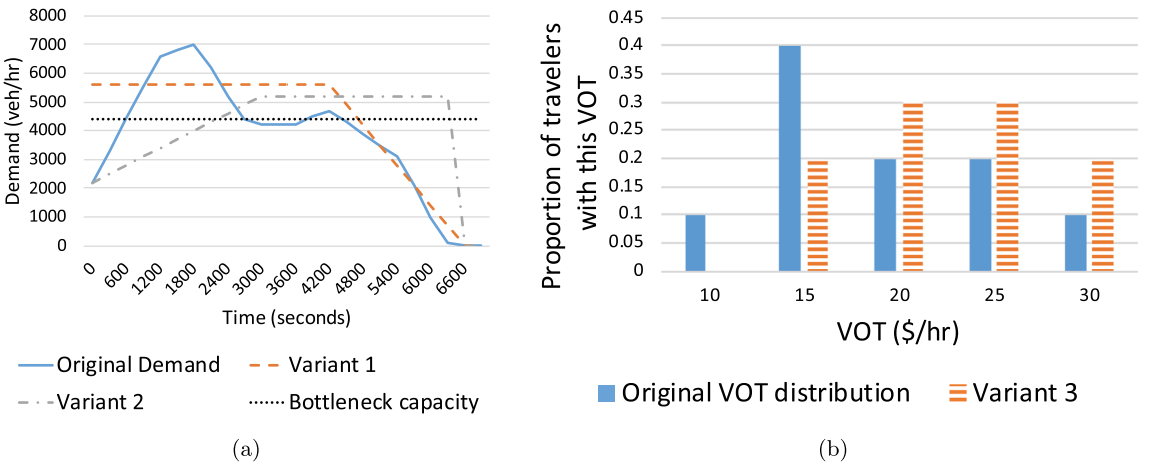


Fig. 6. (a) Demand distributions used for the SESE, DESE and LBJ networks and its variants, and (b) VOT distribution and its variant.

Table 2
Values of parameters used in the simulation

	SESE	DESE	LBJ	MoPac	Parameter	Value
Corridor length (miles)	7.3	1.59	2.91	11.1	$\beta_{\min}$	\$0.1
Simulation duration (hour)	2.5	2	2	3.33	$\beta_{\max}$	\$4.0
$\Delta\tau$ (seconds)	60	300	300	300	q_{ij} (vphpl)	2200
v_{ij} (mph)	55	55	55	65	$k_{\text{jam},ij}$ (veh/mile)	265
σ_o (veh/hr)	50	50	50	50	v_{ij}/w_{ij}	3
σ_d (veh/hr)	10	0	0	100	Δt (seconds)	6

harvest nature of the revenue maximizing profiles. We simulate random toll profiles on the LBJ network and record the congestion profiles for two values of JAH_2: 0.22 and 0.49.[3]

Figs. 7 and 8 show the plots for the time space diagram on managed lane and general purpose lane, and the variation of $\zeta(t)$ for two different toll profiles leading to JAH_2 values of 0.22 and 0.49, respectively. The scale on the time–space diagrams varies from 0, representing no vehicles, to 1, representing jam density. The cell id value on the y-axis is a six-digit number where the first two digits are the tail node of the link, the second two digits are the head node of the link, and the last two digits are the index of the cell number on the link starting from index 1 for the first cell near the tail node. Thus, the increasing value of cell IDs on the y-axis indicates the downstream direction.

As observed, higher value of JAH statistics results in higher congestion on the GPL relative to the ML. When $JAH_2 = 0.22$, vehicles use the ML starting from 1500 s into the simulation. Whereas, when $JAH_2 = 0.49$, vehicles do not enter the managed lane until approximately 2300 s into the simulation, by which the GPLs are heavily congested, indicating more jam-and-harvest behavior.

Table 3 shows values of revenue, TSTT, and JAH_1 for the two toll profiles simulated. We see that the JAH_1 statistic is also high when the JAH_2 statistic is high. The highest revenue is obtained for the highest value of JAH_2 value. TSTT values follow the reverse trend as the revenue: high JAH statistic leads to low TSTT. These experiments help quantify the abstract "jam-and-harvest" nature used in the literature. In Section 4.4, we use reward shaping to generate toll profiles with low JAH_i values ($i = \{1, 2\}$).

4.3. Learning performance of Deep-RL

4.3.1. Learning for different objectives

We next compare the learning performance of the A2C, PPO, and SAC Deep-RL algorithms for both revenue maximization and TSTT minimization objectives. Fig. 9 show the plots of variation of learning for two objectives for all four networks. The plot reports the average objective over 10 random seeds and includes standard deviation as a shaded band around the average.

We make the following observations. First, all Deep-RL algorithms are able to learn "good" objective values, evident in the increasing trend of the average revenue for the revenue maximization objective and a decreasing trend of the average TSTT for the TSTT minimization objective. For the revenue maximization objective, the average revenue values converge to a high value for all networks. For the TSTT minimization objective, the average TSTT values for SESE (Fig. 9b) network did not converge for A2C and PPO algorithms; however a decreasing trend is evident. This lack of convergence is because gradient-based algorithms in stochastic settings may converge to a local optimum or may not converge within desired number of iterations. Therefore, we recommend tracking the value of policy parameters (θ) that achieve the best-found objective over iterations.

Second, we observe that SAC algorithm outperforms A2C and PPO in learning better objectives after simulating less than half of the number of trajectories used by A2C and PPO to reach the same objective level. This better performance is attributed to the entropy regularization used by SAC that promotes active exploration and thus prevents the algorithm getting stuck in a local minimum. Similar to the findings in the literature, SAC is sample-efficient and generates stable solutions across different random seeds (except for the drop in objective value around 400th trajectory for MoPac network). Furthermore, the off-policy deterministic nature of SAC's final policy generates stable revenue or TSTT with lower variance relative to A2C and PPO algorithms that train a stochastic policy. Due to its efficient learning performance, we recommend the use of SAC for dynamic pricing using Deep-RL. There is no evident difference in the performance of A2C and PPO algorithms, except that PPO's learning curve in Fig. 9g is smoother while A2C's continues to oscillate.

Last, in contrast to our expectation that a larger network with high dimensional action space might require large number of iterations to converge, we observe that for both LBJ and MoPac networks, the average objectives converge after simulating less than 200 trajectories. If real-world data is collected every weekday during the peak-hour, we observe that better toll profiles can be learned using data for fewer than 200 weekdays. If the data is obtained from simulation instead, the total average computation time for the SAC algorithm to simulate a trajectory is 1.42 s for the LBJ network and 114.7 s for the MoPac network. This results in less than 5 (40) minutes of simulation for LBJ (MoPac) network to obtain toll profiles with better objectives. Both networks mimic the real-world implementations of managed lanes, and thus we argue that learning is possible within a reasonable number of interactions with the environment even for real-world networks. The amount of data required for training Deep-RL models is often considered its

[3] The JAH_2 values varied between 0.2 and 0.5 for this network as shown in Fig. 12

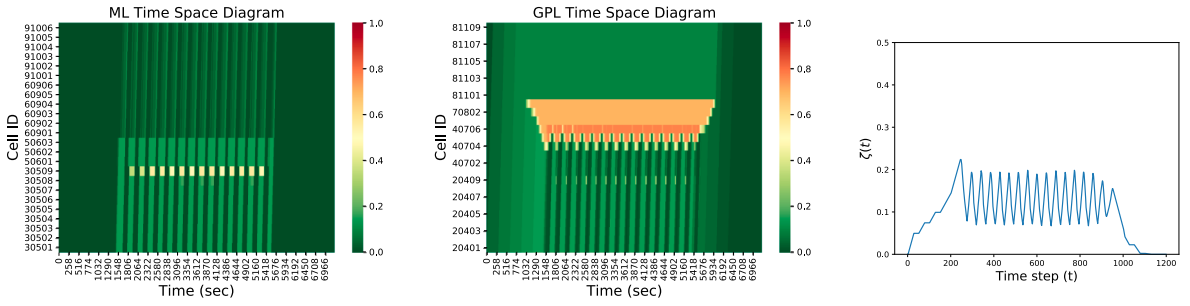


Fig. 7. Plots for $JAH_2 = 0.22$ including (a) the time–space diagram on the ML, (b) the time–space diagram on the GPL, and (c) the variation of statistic $\zeta(t)$ in Eq. (2.5) with .simulation time.

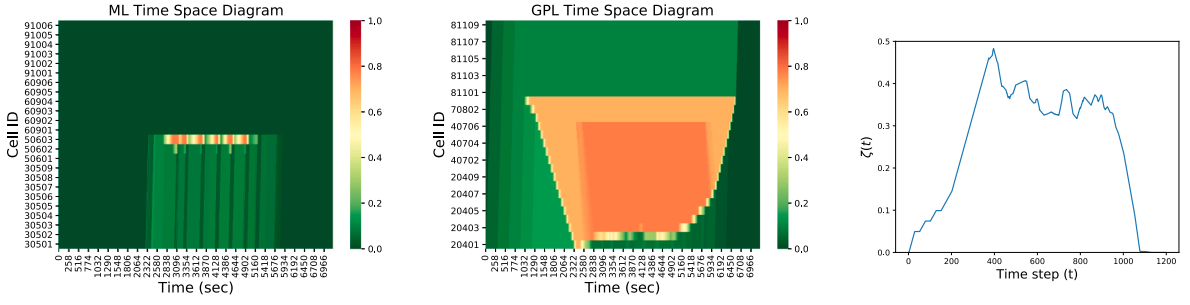


Fig. 8. Plots for $JAH_2 = 0.49$ including (a) the time–space diagram on the ML, (b) the time–space diagram on the GPL, and (c) the variation of statistic $\zeta(t)$ in Eq. (2.5) with .simulation time.

Table 3
Value of different statistics for different cases

Figure	Revenue ($)	TSTT (hr)	JAH_1 (vehicles)	JAH_2
Fig. 7	1203.68	1018.7	451.73	0.22
Fig. 8	4106.03	1421.05	997.23	0.49

major limitation (Arulkumaran et al., 2017); however, for the dynamic pricing problem it is not a constraining factor.

4.3.2. Impact of observation space

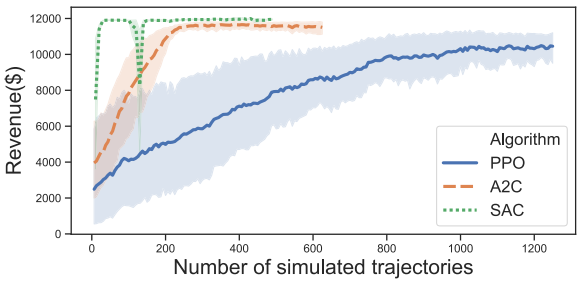
We also test the impact of observation space on the learning of Deep-RL algorithms. For the LBJ network, the results in Fig. 9e and f assumed that flows are observed on all links (which we term `High` observation). We consider two additional observation cases: (a) observing links (3, 5), (4, 7), (6, 9), and (8, 11) (`Medium` observation), and (b) only observing link (6, 9) in the network (`Low` observation). Fig. 10 shows the learning results for revenue maximization objectives for the two algorithms for three levels of observation space.

We observe that changing the observation space has a minor impact on learning rate. This result was unexpected, and suggests that good performance can be obtained with relatively few sensors. We speculate that this happens due to the spatial correlation of the congestion pattern on a corridor (where observing additional links does not add a new information for setting the tolls). The computation time differences on using different observation spaces were also not significant.

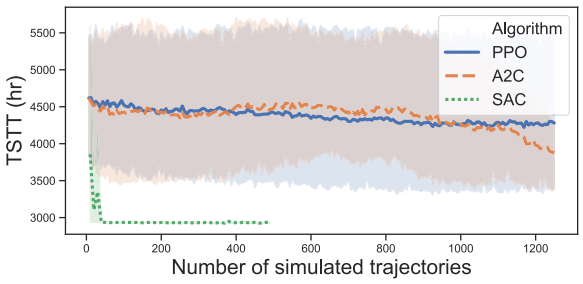
These findings indicate that a toll operator can learn toll profiles optimizing an objective without placing sensors on all links, which is a lower cost alternative than observing all links. Future work will be devoted to the cost-benefit analysis of different sensor-location combinations assuming variability in sensing errors across different sensors. (FW#5)

4.3.3. Learning for varied inputs and transferability analysis

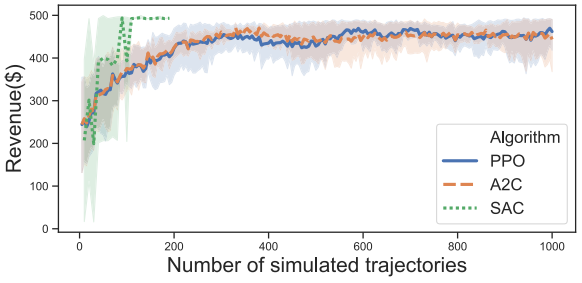
In this section, we consider how Deep-RL algorithms perform for varied set of inputs and how the policies trained on one set of inputs perform when transferred to new inputs without retraining for the new inputs. This analysis is useful for a toll operator who trains the algorithm in a simulation environment for certain assumptions of input. For the policy to transfer, the observation space in the new setting must be identical to the setting where the transferred policy is trained. We only consider cases for changes in input



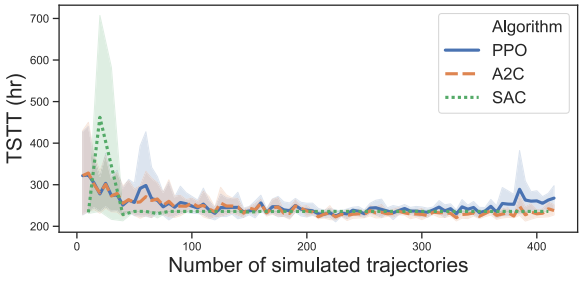
(a) SESE Revenue Maximization.



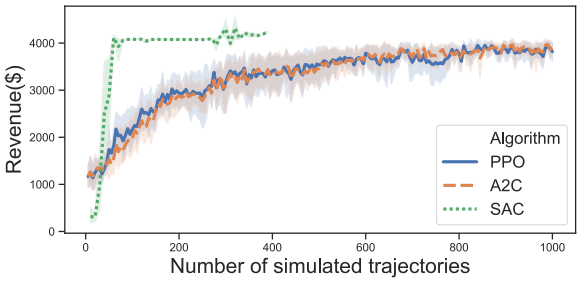
(b) SESE TSTT Minimization.



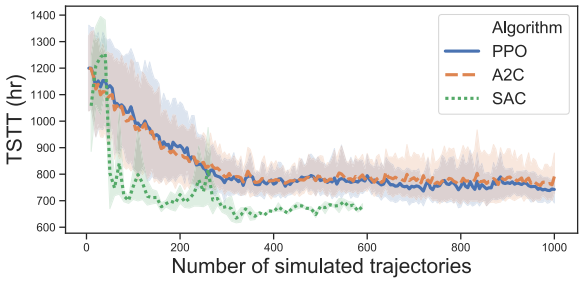
(c) DESE Revenue Maximization.



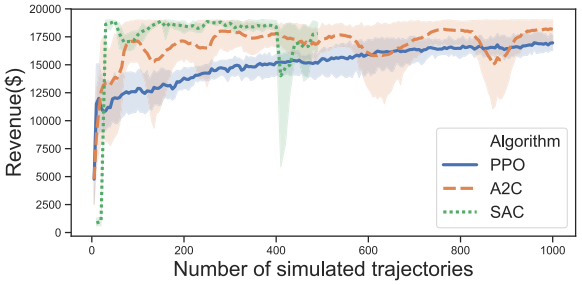
(d) DESE TSTT Minimization.



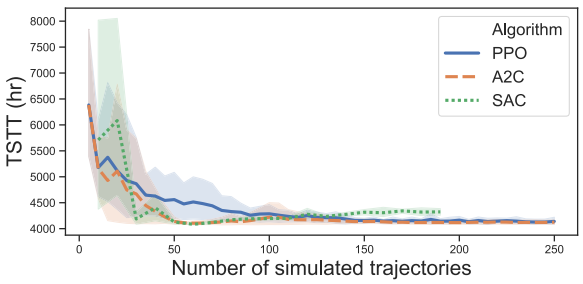
(e) LBJ Revenue Maximization.



(f) LBJ TSTT Minimization.



(g) MoPac Revenue Maximization.



(h) MoPac TSTT Minimization.

Fig. 9. Plot of average objective value and the confidence interval with the number of simulated trajectories over 10 random seeds for the four networks.

demand distribution, VOT distribution, and lane choice model. Transferability of Deep-RL algorithms trained on one network to other networks or the same network with new origins and destinations requires extensive investigation and is a topic for future research (FW#6).

We consider the revenue-maximizing policy for the LBJ network and consider four different input cases. The first two cases consider new demand distributions (Variant 1 and Variant 2) shown in Fig. 6a. The third case considers a new VOT distribution (Variant 3) shown in Fig. 6b. And, the last case uses a multiclass binary logit model with scaling parameter 6 for modeling driver lane

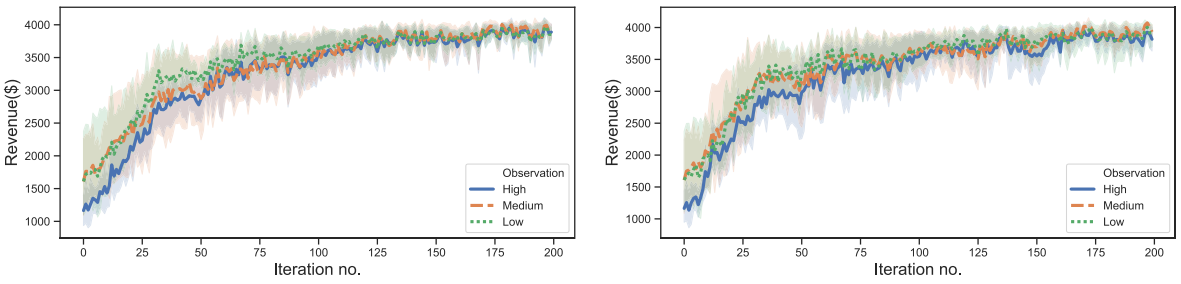


Fig. 10. Plot of the average revenue with iteration over 5 random seeds for the three levels of observation for (a) A2C algorithm, and (b) PPO algorithm for the LBJ network.

choice (Pandey and Boyles, 2019). For each case, we also directly apply the policy obtained at the final iteration of training on the LBJ network for the revenue-maximization objective with the original demand, VOT distribution, and lane choice model (Fig. 9e).

Fig. 11 show the plots of variation of revenue with iterations while learning from scratch for both A2C and PPO algorithms and the average revenue (and its full range of variation) obtained from the transferred policy for the new inputs. The average is reported over 100 runs of the transferred policy for new inputs without retraining.

First, we observe that learning for the new input configurations "converges" after simulating 500 trajectories for all four cases. This observation indicates that the Deep-RL algorithms can iteratively learn "good" toll profiles regardless of the input distribution. This is a significant advantage over the MPC-based algorithms in the literature that require assumptions on driver behavior and inputs to solve the optimization problem at each time step. Similar to the previous cases, both A2C and PPO algorithms perform almost identically with less than ~10% difference in the objective values for the four cases. This is in contrast to the other environments used for testing Deep-RL algorithms like Atari games and `MuJoCo` where the PPO algorithm is significantly better than the A2C algorithm (Schulman et al., 2017). This is because the state update in the ML pricing problem is not drastically influenced by the toll actions, unlike the high uncertainty in the state transition in the Atari and `MuJoCo` environments. Thus, the A2C algorithm does not produce large-policy updates and has no relative disadvantage over the PPO algorithm, explaining their almost-identical performance.

Second, the average revenue of the transferred policy is within $5 - 12\%$ of the average revenue at termination while learning from scratch. For case 3 with VOT variant, the transferred policy does even better than the policy learned from scratch after 100 iterations of training. The observations from the first three cases suggest that even though the Deep-RL algorithms were not trained for the new

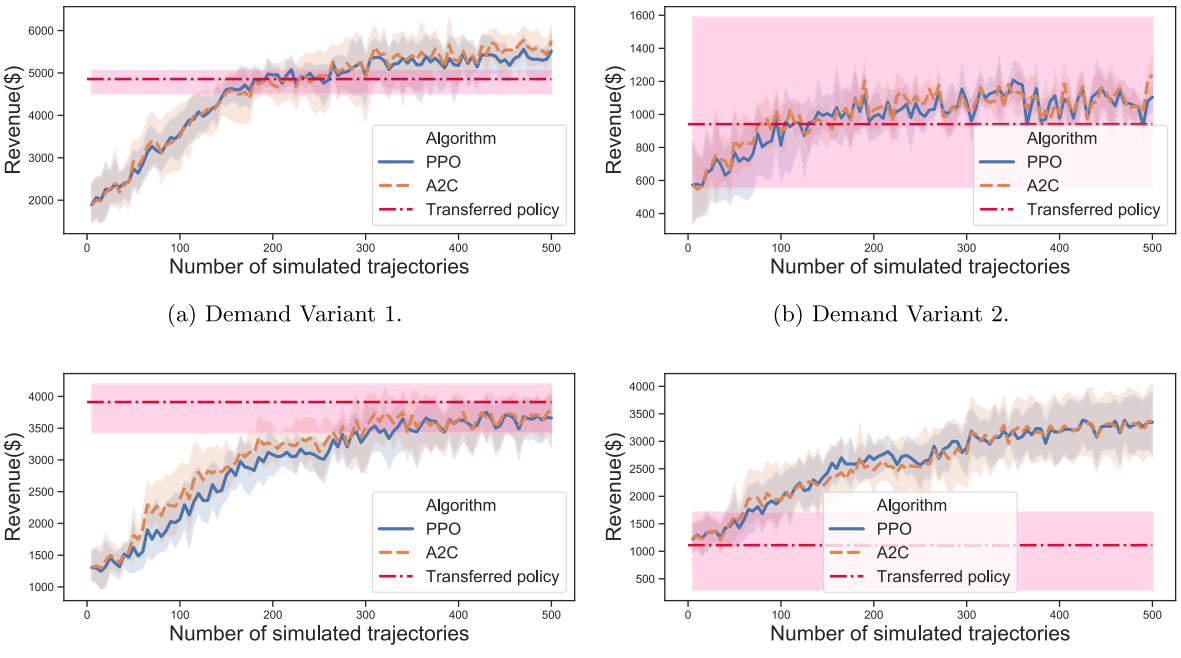


Fig. 11. Comparing learning-from-scratch performance of the A2C and PPO algorithms on different input distributions with the policy transferred after learning on the original distribution (shown as a horizontal line-dot pattern) for the LBJ network.

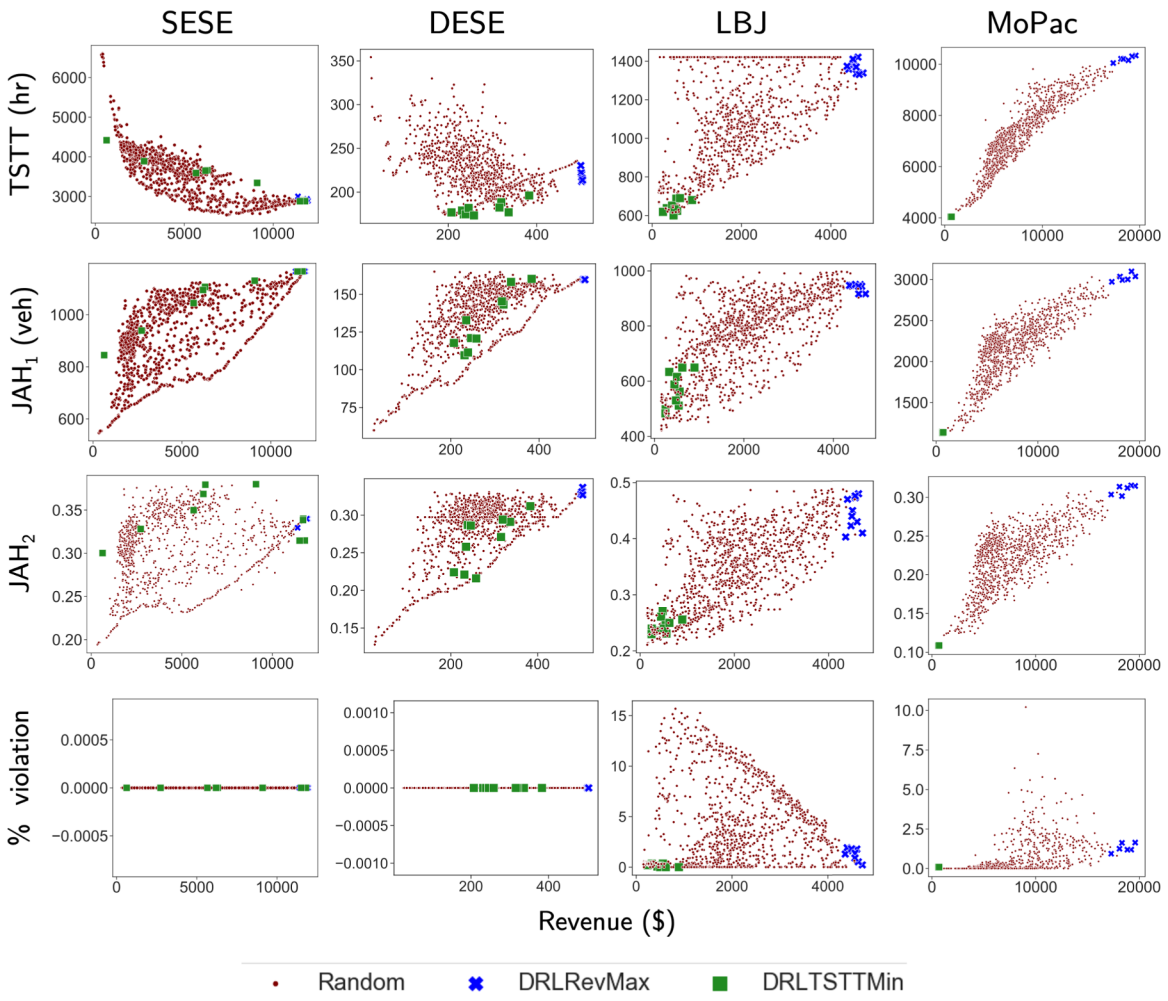


Fig. 12. Plot of various objectives against the revenue for 1000 randomly generated toll profiles (Random) and the profiles generated from Deep-RL for revenue maximization (DRLRevMax) and TSTT minimization (DRLTSTTMin) objectives.

inputs, they are able to learn characteristics of the congestion in the network and perform well (on an average) on the new inputs. However, for case 2, the transferred policy has a lot of variance in the generated revenue; this is because small changes in input tolls have higher impact on generated revenue for demand Variant 2.

Third, contrary to the first three cases, the transfer of policy in case 4 did not work well: the average revenue of transferred policy is 40% of the maximum revenue obtained. This is because the multiclass logit model predicts significantly different proportion of splits of travelers at a diverge and thus have a significant impact on the evolution of congestion. Both cases 3 and 4 impact the split of travelers at the diverge, yet the performance of transferred policy is very different for both cases. This finding suggests that the driver lane choice model should be carefully selected and calibrated for Deep-RL training for reliable transfer to the real-world environments, whereas the demand and VOT distributions are less important.

4.4. Multi-objective optimization

We next focus our attention on multiple optimization objectives together. In the literature, revenue maximization and TSTT minimization objectives are shown to be conflicting (Pandey and Boyles, 2018a), that is toll policies generating high revenue have a high value of TSTT. Finding toll profiles that satisfy both objectives to a degree is the focus of this section.

We consider how different objectives vary with respect to each other for 1000 randomized toll profiles simulated for all four networks. Fig. 12 shows the plots of variation of TSTT, JAH_1, JAH_2, and `%-violation` against the revenue obtained from the toll policies. Because all vehicles exit at the end of simulation, throughput for all toll profiles is a constant equal to the number of vehicles using the network. The figure also shows the values of objectives from the toll profiles generated by Deep-RL algorithms where "DRLRevMax" indicates toll profiles from the revenue maximization objectives and "DRLTSTTMin" indicates toll profiles from the TSTT minimization objective.

We make following observations:

1. First, the best toll profiles generated from Deep-RL algorithm are the best found among the other randomly generated profiles for the respective objectives (except for TSTT minimization on SESE network where the A2C and PPO algorithms did not converge after simulating 1200 trajectories).
2. Second, similar to the trends in the literature, toll profiles generating high revenue also generate high values of TSTT for the LBJ and MoPac networks. However, for the SESE and DESE networks, the trend does not hold. This behavior, where revenue-maximizing tolls do not differ significantly from the TSTT-minimizing tolls is possible for networks where GPLs are jammed quickly enough. Once the GPL is jammed, revenue is maximized by charging the highest possible toll while sending maximum number of vehicles towards the ML. Such tolls will also generate low values of TSTT as they utilize the ML to its full capacity from that time step onwards. This finding indicates that, depending on the network properties and the inputs, the two objectives may not always be in conflict with each other. We leave a detailed analysis of how different network characteristics impact the differences between revenue-maximizing and TSTT-minimizing tolls for future work (FW#7).
3. Third, we see that tolls generating high revenue also have high values of JAH_1 and JAH_2 statistics. The tolls generating low TSTT, however, do not have a fixed trend. For example, for the MoPac network, tolls generating low TSTT have lower revenue and thus have lower values of JAH statistics; however, for the other networks, JAH statistics are also relatively high for the tolls minimizing TSTT compared to the least JAH statistic value obtained. This finding shows that tolls minimizing TSTT may also exhibit JAH behavior, though the extent of JAH for TSTT-minimizing profiles is always lower than the revenue-maximizing profiles.
4. Last, for the LBJ and MoPac networks with multiple access points to the ML, we observe that several toll profiles can cause violation of the speed limit constraint. However, the toll profiles optimizing the revenue or TSTT generate `%-violation` less than 2% for both MoPac and LBJ networks.

Next, we seek toll profiles that optimize two objectives. Multi-objective reinforcement learning is an area that focuses on the problem of optimizing multiple objectives (Liu et al., 2014). There are two broad approaches for solving this problem: single-policy approach and multi-policy approach. Single-policy approaches convert the multi-objective problem into a single objective by defining certain preferences among different objectives like defining a weighted combination of multiple objectives. Multi-policy approaches seek to find the policies on the Pareto frontier of multi objective. In this article, we focus on the single-policy approach due to its simplicity. We consider the weighted-sum and threshold-penalization approaches explained next.

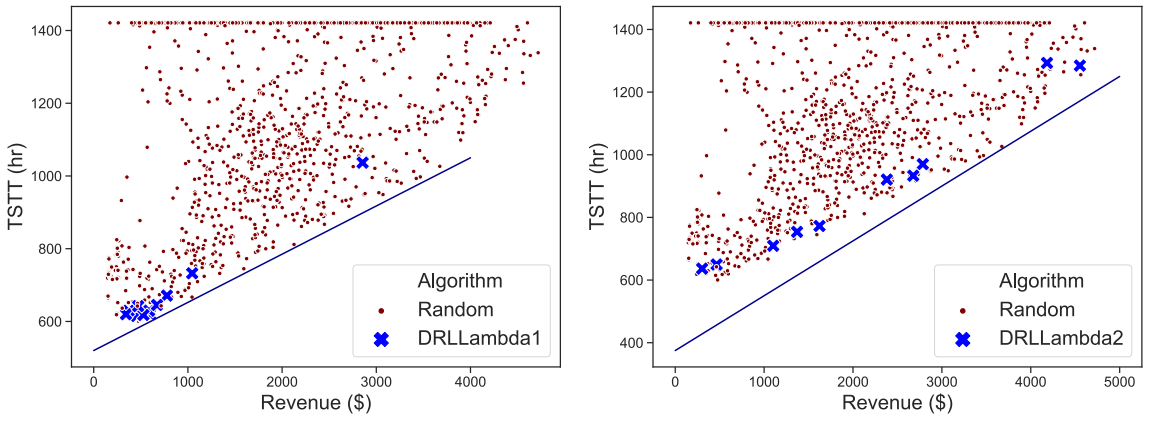
First, we apply the weighted-sum approach for finding a single policy that jointly optimizes TSTT and revenue. We define a new joint reward function $r^{\text{joint}}(s, a)$ as a linear combination of two rewards:

$$r^{\text{joint}}(s, a) = \lambda r^{\text{RevMax}}(s, a) + r^{\text{TSTTMin}}(s, a). \tag{4.1}$$

The value of λ is the relative weight of revenue ($) with respect to TSTT (hrs) and has units hr/"$". Geometrically, λ represents the slope of a line on the TSTT-Revenue plot.

We run A2C and PPO algorithms for the new reward on the LBJ network with two different values of λ: $\lambda_1 = 0.1325$ hr/$ and $\lambda_2 = 0.175$ hr/$ (the values are chosen so that toll profiles in the mid-region of the TSTT-revenue plot are potentially optimal). Fig. 13 shows the plot of optimal toll profiles obtained from Deep-RL algorithms on the TSTT-Revenue space. The slopes of the lines, equal to the λ values, are also shown, and the lines are positioned by moving them from the bottom to the top till they touch the first point among the generated space of points (that is, the line is approximately a tangent to the Pareto frontier).

As observed, Deep-RL algorithms are able to learn toll profiles that maximize the joint reward. For the λ_1 case, toll profiles are generated very close to the Pareto frontier; however, they are concentrated in the region where both TSTT and revenue are lower indicating the presence of local minima in the region. For the λ_2 case, the toll profiles are more spread out in terms of their values of



(a) TSTT vs Revenue λ_1. (b) TSTT vs Revenue λ_2.

Fig. 13. Plot of TSTT vs revenue for the LBJ network for toll profiles generated randomly and toll profiles generated after optimizing the joint reward for two different values of λ.

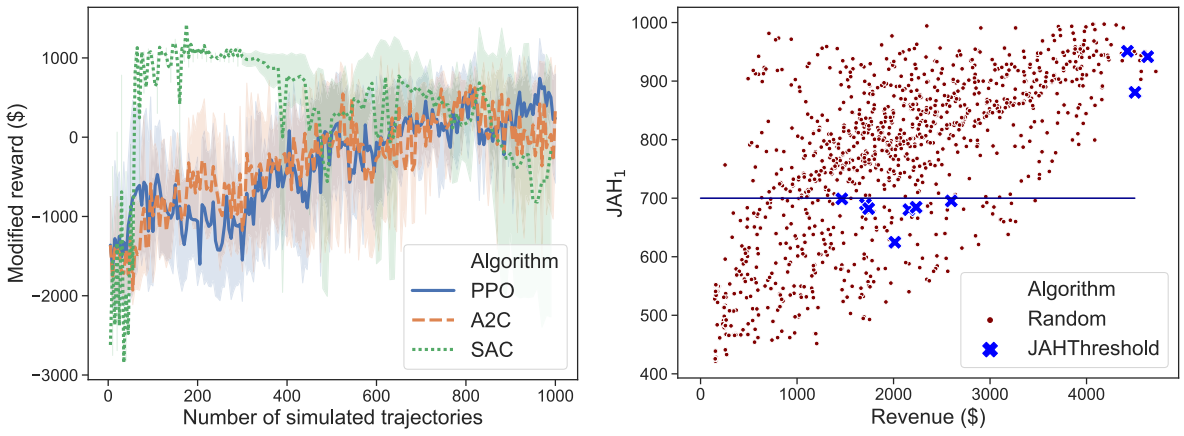


Fig. 14. (a) Plot of average modified reward with iteration while maximizing revenue with a reward penalty of − "$"3000 if the JAH_1 statistic is more than 700 vehicles, and (b) the plot of JAH_1 vs revenue for the best-found toll profiles from the threshold-penalization method, along with toll profiles generated randomly.

TSTT and revenue; however, there are still a few toll profiles that are closer to the Pareto frontier tangent line which the Deep-RL algorithms did not find. This can again be explained by the behavior of policy gradient algorithms which are prone to converge to local optimum because they follow a gradient-ascent approach.

Optimizing using a joint reward definition as Eq. (4.1) can also be interpreted as following: that a toll operator is willing to sacrifice "$"1 revenue for a $1/\lambda$ hours decrease in TSTT value. For the two values of λ, λ_1 and λ_2, this is equivalent to sacrificing "$"1 revenue for a 7.55 hours and 5.72 hours decrease in total delay for the system, respectively. If they trade off these objective outside this range, the optimal policy will be the same as solely maximizing revenue or minimizing TSTT.

The second approach for solving multi-objective optimization problem is the threshold approach where we find toll policies that maximum revenue (minimize TSTT) such that TSTT (revenue) is less (higher) than a certain threshold. In this article, we apply the threshold-penalization method to model threshold constraints. This method simulates a policy and if at the end of an episode the constraint is violated, a high negative value is added to the reward to penalize such update. We test this technique to find tolls that maximize revenue such that JAH_1 statistic is less than a threshold value. We use JAH_1 statistic because it has a physical interpretation and, unlike JAH_2, is not unitless.

We conduct tests for the threshold-penalization technique on the LBJ network with a threshold JAH_1 of 700 vehicles and add a reward value of − "$"3000 to the final reward if at the end of simulation the JAH_1 statistic is higher than the threshold. Fig. 14a shows the learning curve plotting the variation of modified reward with iterations. We observe that each of A2C, PPO, and SAC algorithms improve the modified reward with iterations with the expected better performance from SAC algorithm; however, it is hard to argue that they have converged. Learning better toll profiles while using threshold-penalization method is difficult due to the *credit assignment problem* where it is unclear which toll over an episode resulted in the constraint violation. Fig. 14b shows the plot for tolls obtained from threshold-penalization technique (which are the best case tolls obtained by PPO algorithm) on the JAH_1-Revenue space.

As observed, the threshold-penalization method is able to learn toll profiles with desired JAH value for 7 out of 10 random seeds. However, the learned toll profile is not the best found (that is, there are toll profiles with JAH less than 700 but generating revenue higher than "$"2800, which is the best found revenue). This is because the modified reward did not converge (yet) after 200 iterations. Despite the lack of convergence, we conclude that the penalization method is a useful tool to model constraints on toll profiles. The success of threshold-penalization method depends on the random seed, as that determines which local minimum the algorithm will converge to. While the violation of speed limit constraint was not significant in our experiments, we also recommend the use of threshold-penalization method if the `%-violation` statistic is higher that desired for the optimal profiles.

4.5. Comparison with other heuristics

In this section, we compare performance of the Deep-RL algorithms against other methods.

For the PID controller we calibrate the objective for different values of η, K_p, K_i, and K_d terms in Eq. (3.12). We consider η values between 0 and 1, and tune the K_p, K_i, and K_d values heuristically by evaluating their performance for different combinations. We consider following sets for calibration: $K_p \in [0.0{:}0.05{:}0.5]$, $K_d = K_p T_d$ and $K_i = K_p / T_i$. T_i and T_d represent the integration time and the derivative time, respectively and are sampled between 5 min and 3 h in increments of 15 min. This choice allows us to scale the magnitudes of terms being multiplied to K_i and K_d appropriately.

In our simulations, we observe that low values of η generate the highest average revenue across all combinations. Lower values of η ensure that ML is kept relatively more congestion free than the case when η value is high. A low value of η charges high toll in the

Table 4
Comparison of Deep-RL against the PID controller for the two optimization objectives. For Deep-RL, we report the better average objective among A2C or PPO across 20 random initializations, whereas for the PID controller, we report the better average objective across different values of calibration parameters. Values in bold are statistically significant compared using a two-sample t-test.

	Deep-RL	PID Controller
Revenue Maximization		
SESE	**$11981.80 ± 44.49** **SAC**	$11874.84 ± 21.44 $\eta = 0; K_p = 0.25; K_i = 0; K_d = 0$
DESE	**$503.52 ± 2.32** **SAC**	$480.54 ± 23.20 $\eta = 0; K_p = 0.1; K_i = 7E-6; K_d = 30$
LBJ	**$4572.91 ± 53.07** **PPO**	$4215.99 ± 290.66 $\eta = 0; K_p = 0.1; K_i = 0; K_d = 0$
MoPaC	**$19161.17 ± 34.47** **SAC**	$18874.10 ± 94.68 $\eta = 0; K_p = 0.4; K_i = 0.0013; K_d = 240$
TSTT minimization objective		
SESE	**2892.02 ± 11.52 h** **SAC**	2932.11 ± 54.27 h $\eta = 0.2; K_p = 0.01; K_i = 0; K_d = 0$
DESE	**178.65 ± 5.74 h** **A2C**	198.87 ± 6.68 h $\eta = 0.3; K_p = 0.01; K_i = 1E-6; K_d = 36$
LBJ	**624.33 ± 16.54 h** **SAC**	661.40 ± 0 $\eta = 1; K_p = 0.1; K_i = 0; K_d = 0$
MoPaC	**4039.87 ± 3.15** **PPO**	4075.20 ± 4.68 $\eta = 1; K_p = 0.25; K_i = 0; K_d = 0$

beginning and ensures that GPLs are more jammed promoting jam-and-harvest nature and generating more revenue.

In contrast to this, low values of TSTT are obtained for high values of η for both LBJ and MoPac networks. This is also intuitive: tolls minimizing TSTT operate the managed lane close to its critical density at all times. The impact of K_p, K_i and K_d terms on the revenue and TSTT has no consistent trend and varies with the network.

Table 4 shows the best-case performance for the algorithms for both the revenue maximization and TSTT-minimization objectives. For the Deep-RL algorithm, we report the better average objective value among A2C, PPO, and SAC algorithms, and the corresponding standard deviation across 20 random seeds. For the PID controller, we report the best average objective value across different values of calibration parameters. The standard deviation for the PID controller is reported over 20 different random toll initialization at the initial step $k = 0$. We also compare the statistical significance of the higher value by checking the hypothesis that the means of the two algorithms are not equal using a two-sample t-test with a 5% level of significance and highlight the statistically significant result in bold.

We observe that the Deep-RL algorithms always find tolls with better objective values compared to the feedback control heuristic. For the revenue-maximization objective, the average revenues from Deep-RL are 0.9–8.5% higher than the ones obtained from the PID controller. Higher revenues generated by Deep-RL algorithm in all cases are also statistically significant. Similarly, for the TSTT-minimization objective, Deep-RL algorithm find tolls with better objective for all networks. The average TSTT values obtained from the Deep-RL algorithm are 0.8–8.4% lower than the average TSTT from the PID controller. We find that SAC algorithm learns the highest average objective in 5 out of 8 cases. However, the differences between the average objective values for Deep-RL algorithms are not statistically significant.

With average revenue and TSTT within 10% of the best-found objective, we find that the PID controller provides a good approximation of tolls obtained from Deep-RL training for both objectives. The PID controller also has a computational benefit as it only requires a one shot calculation based on the real-time observations. Nevertheless, the Deep-RL framework is still advantageous because of its benefit of transferability to new input domains and the capability of multi-objective optimization. Future work can be devoted in devising other heuristics that combine the optimization efficiency of Deep-RL algorithms and the computational efficiency of feedback control heuristics (FW#8).

Last, we also compare the performance of Deep-RL against VFA and SparseV algorithms which assume full observability. These algorithms rely on look-up table representation of value functions and discretize the state space.[4] For uniform comparison with the previous studies which use distance-based tolling (Pandey and Boyles, 2018a,b), we conduct tests only on DESE and LBJ networks with toll values varying between "$"0.05/mile and "$"0.8/mile. Additionally, since the SparseV algorithm in Pandey and Boyles (2018b) is only defined for the revenue-maximization objective, we only focus on that objective. Table 5 shows the best found revenue for the two networks using the three algorithms. As observed, Deep-RL algorithm outperforms VFA and SparseV by generating an average 11.85% percent higher revenue. The performance of VFA is especially poor for the LBJ network due to high-

[4] It is possible to implement neural networks as function approximators in VFA and SparseV than using look-up tables; however, our focus is on direct comparison with previous algorithms in the literature.

Table 5
Comparison of best-found toll objective from Deep-RL algorithm with partial observability against the VFA and SparseV heuristics that assume full observability

	Revenue-maximization best-found revenue ($)		
	Deep-RL	VFA	SparseV
DESE	503.71	500.18	493.80
LBJ	4634.69	2880.59	4316.03

dimensional action space where VFA is unable to learn the desired coordination between different toll gantries for maximizing revenue. These findings show that even under partial observability Deep-RL algorithms can learn toll profiles with better objectives than algorithms assuming full observability.

5. Conclusion

In this article, we developed Deep-RL algorithms for dynamic pricing of MLs with multiple access points. We showed that the Deep-RL algorithms are able to learn toll profiles for multiple objectives, even capable of generating toll profiles lying on the Pareto frontier. The average objective value converged within 200 iterations for the four networks tests. The number of sensors and sensor locations were found to have little impact on the learning due to the spatial correlation of congestion pattern. We also conducted transferability tests and showed that policies trained using Deep-RL algorithm can be transferred to setting with new demand distribution and VOT distribution without losing performance; however, if the lane choice model is changed the transferred policy performs poorly. We analyzed the variation of multiple objectives together and found that TSTT-minimizing profiles may be similar to revenue-maximizing profiles for certain network characteristics where the GPL invariably becomes congested early in the simulation. We also compared the performance of Deep-RL algorithms against the feedback control heuristic and found that it outperformed the heuristic for the revenue maximization objective generating average revenue up to 8.5% higher than the heuristic and generating average TSTT up to 8.4% lower than the heuristic.

The Deep-RL model in this article requires training, which is dependent on the input data and the parameters. We make following implementation recommendations. If a toll operator has access to the input data including the demand distribution and driver lane choice behavior, we recommend first calibrating a lane-choice model using the data and then using the calibrated model to train the policy for the desired objective under desired constraints. If the driver lane choice data is very detailed and can exactly identify how many travelers chose the ML at each time, then that data can be directly used in training without calibrating a lane-choice model; however, a calibrated model is still recommended as it can assist in conducting sensitivity analysis to other inputs and/or long-term planning. If the input data is not available or has poor accuracy, we recommend two alternatives. A toll operator can either train the Deep-RL model considering high stochasticity by choosing a large values for the standard deviations (σ_d and σ_o), or train several policies for different combinations of inputs and use the policy based on the expected realization of inputs from field data for real-time implementation. Lastly, we also recommend retraining the toll policy using real-time data. For example, a policy can be trained from the historic data and then improved based on the observations from a specific day and the improved policy can then be applied to the next day. Additionally, though the model in this article trains a stochastic policy, for implementation purposes, we can use a deterministic policy with the tolls set as the mean value predicted by the policy.

In addition to the future work ideas discussed earlier (marked as FW#), there are additional topics that should be studied. First, the choice of traffic flow model is critical to the performance of Deep-RL algorithms. The macroscopic multiclass cell transmission model used in our analysis does not capture the impacts of lane changes and the second-order stop-and-go waves. Future work can be devoted to developing efficient Deep-RL algorithm using microscopic simulation models and on testing the transferability of algorithms trained on a macroscopic scale to microscopic scales. Second, we only considered loop detector density measurements in the simulations. Other types of observations like speeds, toll-tag readings, and measurements using Lagrangian sensors like GPS devices on vehicles require redefining the POMDP to handle such measurements and can be looked into as part of the future work. Third, for real-time implementation of Deep-RL algorithms, the minimum speed limit constraint on ML (constraint 2 defined in Section 2.4) should be satisfied throughout the learning phase, which requires analysis of constrained policy optimization methods like in Achiam et al. (2017) and techniques from safe reinforcement learning (Garcıa and Fernández, 2015). Last, the future work should also analyze the equity impacts of the tolls generated by Deep-RL across multiple vehicle classes and investigate if generating equitable toll policies can be included as part of the Deep-RL problem.

CRediT authorship contribution statement

Venktesh Pandey: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Visualization, Writing - original draft, Writing - review & editing. **Evana Wang:** Data curation, Investigation, Visualization. **Stephen D. Boyles:** Conceptualization, Funding acquisition, Supervision, Writing - review & editing.

Acknowledgment

Partial support for this research was provided by the North Central Texas Council of Governments, Data-Supported Transportation Planning and Operations University Transportation Center, and the National Science Foundation (Grants No. 1254921, 1562291, and 1826230.) The authors are grateful for this support. The authors would also like to thank Natalia Ruiz Juri and Tianxin Li at the Center for Transportation Research, The University of Texas at Austin for their help in providing us the data for the MoPac Express lanes, and Josiah Hanna for his comments on the paper draft.

Appendix A. Hyperparameters

A.1. A2C and PPO

Table 6 shows the hyperparameters used for producing plots of A2C and PPO for Figs. 9, 10, 11, and 14a.

Table 6
A2C and PPO hyperparameters.

	SESE	DESE	LBJ	MoPac
Non-linearity		tanh		
Number of hidden units per layer		64		
Number of hidden layers		2		3
Learning rate for policy update		3×10^{-4}		3×10^{-3}
Learning rate for value function updates		10^{-3}		
Number of iterations for value function updates		80		
γ^{GAE}		0.99		
λ^{GAE}		0.97		
Replay buffer size for GAE	600	120	120	400
Number of episodes between each update for policy and value function parameters		5		
Gradient descent/ascent optimizer		Adam (Kingma and Ba (2014))		
Standard deviation of Gaussian policy		0.6065		

A.2. SAC

Table 7 shows the hyperparameters used for producing plots of SAC for Figs. 9 and 14a.

Table 7
SAC hyperparameters.

	SESE	DESE	LBJ	MoPac
Non-linearity		ReLU		
Number of hidden units per layer		256		
Number of hidden layers		2		
Learning rate		10^{-3}		
Entropy regularization coefficient		0.2		
Replay buffer size		10^6		
Target network smoothing coefficient		0.005		
Number of samples per minibatch		256		
Number of episodes to wait before starting update		2		
Update after every		1 episode		
Gradient descent/ascent optimizer		Adam (Kingma and Ba, 2014)		
Number of episodes for testing the deterministic policy		10		

References

Åström, K.J., Murray, R.M., 2010. Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press.
Achiam, J., Held, D., Tamar, A., Abbeel, P., 2017. Constrained policy optimization. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 22–31 JMLR.org.
Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. 34 (6), 26–38.
Belletti, F., Haziza, D., Gomes, G., Bayen, A.M., 2017. Expert level control of ramp metering based on multi-task deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 19 (4), 1198–1207.

Burris, M.W., Brady, J.F., 2018. Unrevealed preferences: unexpected traveler response to pricing on managed lanes. Transp. Res. Rec. 2672 (5), 23–32.
Chu, T., Wang, J., Codecà, L., Li, Z., 2019. Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans. Intell. Transp. Syst.
Daganzo, C.F., 1995. The cell transmission model, part II: Network traffic. Transp. Res. Part B: Methodol. 29 (2), 79–93.
Dorogush, E.G., Kurzhanskiy, A.A., 2015. Modeling toll lanes and dynamic pricing control. arXiv:1505.00506.
Ferrara, A., Sacone, S., Siri, S., 2018. Freeway Traffic Modelling and Control. Springer.
Garcıa, J., Fernández, F., 2015. A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. 16 (1), 1437–1480.
Gardner, L.M., Bar-Gera, H., Boyles, S.D., 2013. Development and comparison of choice models and tolling schemes for high-occupancy/toll (HOT) facilities. Transp. Res. Part B: Methodol. 55, 142–153.
Gardner, L., Boyles, S.D., Bar-Gera, H., Tang, K., 2015. Robust tolling schemes for high-occupancy/toll (HOT) facilities under variable demand. Transp. Res. Rec. 2450, 152–162.
Genders, W., Razavi, S., 2016. Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142.
Göçmen, C., Phillips, R., van Ryzin, G., 2015. Revenue maximizing dynamic tolls for managed lanes: A simulation study.
Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290.
Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
LBJ, 2016. LBJ express FAQs. http://www.lbjtexpress.com/faq-page/t74n1302. Last Accessed: June 20, 2019.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.
Liu, C., Xu, X., Hu, D., 2014. Multiobjective reinforcement learning: a comprehensive overview. IEEE Trans. Syst. Man Cybernet.: Syst. 45 (3), 385–398.
Lou, Y., Yin, Y., Laval, J.A., 2011. Optimal dynamic pricing strategies for high-occupancy/toll lanes. Transp. Res. Part C: Emerg. Technol. 19 (1), 64–74.
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
n.a., 2020. Managed Lanes Project Database. https://managedlanes.wordpress.com/category/projects/. Last Accessed: January 20, 2020.
OpenAI, 2019. Welcome to Spinning Up in Deep RL– Spinning Up documentation. https://spinningup.openai.com/en/latest/index.html. Last Accessed: June 20, 2019.
Pandey, V., 2016. Optimal dynamic pricing for managed lanes with multiple entrances and exits. Master's thesis. The University of Texas at Austin.
Pandey, V., Boyles, S.D., 2018a. Dynamic pricing for managed lanes with multiple entrances and exits. Transp. Res. Part C: Emerg. Technol. 96, 304–320.
Pandey, V., Boyles, S.D., 2018b. Multiagent reinforcement learning algorithm for distributed dynamic pricing of managed lanes. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 2346–2351.
Pandey, V., Boyles, S.D., 2019. Comparing route choice models for managed lane networks with multiple entrances and exits. Transp. Res. Rec. 2673 (10), 381–393.
Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Rezende, D.J., Badia, A.P., Vinyals, O., Heess, N., Li, Y., et al., 2017. Imagination-augmented agents for deep reinforcement learning. Adv. Neural Inf. Process. Syst. 5690–5701.
Schulman, J., 2016. Optimizing expectations: From deep reinforcement learning to stochastic computation graphs. PhD thesis. UC Berkeley.
Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P., 2015. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Shabestary, S.M.A., Abdulhai, B., 2018. Deep learning vs. discrete reinforcement learning for adaptive traffic signal control. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 286–293.
Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.
Tan, Z., Gao, H.O., 2018. Hybrid model predictive control based dynamic pricing of managed lanes with multiple accesses. Transp. Res. Part B: Methodol. 112, 113–131.
Toledo, T., Mansour, O., Haddad, J., 2015. Simulation-based optimization of HOT lane tolls. Transp. Res. Procedia 6, 189–197.
van der Pol, E., 2016. Deep reinforcement learning for coordination in traffic light control. Master's thesis. University of Amsterdam.
Van der Pol, E., Oliehoek, F.A., 2016. Coordinated deep reinforcement learners for traffic light control. In: Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016).
Wu, C., Kreidieh, A., Parvate, K., Vinitsky, E., Bayen, A.M., 2017. Flow: Architecture and benchmarking for reinforcement learning in traffic control. arXiv preprint arXiv:1710.05465.
Yang, L., Saigal, R., Zhou, H., 2012. Distance-based dynamic pricing strategy for managed toll lanes. Transp. Res. Rec.: J. Transp. Res. Board 2283, 90–99.
Yau, K.-L.A., Qadir, J., Khoo, H.L., Ling, M.H., Komisarczuk, P., 2017. A survey on reinforcement learning models and algorithms for traffic signal control. ACM Comput. Surveys (CSUR) 50 (3), 34.
Zhang, Y., Atasoy, B., Ben-Akiva, M., 2018. Calibration and optimization for adaptive toll pricing. In: 2018 97th Annual Meeting of Transportation Research Board, pp. 18–05863. TRB.
Zhu, F., Ukkusuri, S.V., 2015. A reinforcement learning approach for distance-based dynamic tolling in the stochastic network environment. J. Adv. Transp. 49 (2), 247–266.