Learning How to Dynamically Route Autonomous Vehicles on Shared Roads

Daniel A. Lazar*, Erdem Bıyık*, Dorsa Sadigh, Ramtin Pedarsani

***Abstract*—Road congestion induces significant costs across the world, and road network disturbances, such as traffic accidents, can cause highly congested traffic patterns. If a planner had control over the routing of all vehicles in the network, they could easily reverse this effect. In a more realistic scenario, we consider a planner that controls autonomous cars, which are a fraction of all present cars. We study a dynamic routing game, in which the route choices of autonomous cars can be controlled and the human drivers react selfishly and dynamically. As the problem is prohibitively large, we use deep reinforcement learning to learn a policy for controlling the autonomous vehicles. This policy indirectly influences human drivers to route themselves in such a way that minimizes congestion on the network. To gauge the effectiveness of our learned policies, we establish theoretical results characterizing equilibria and empirically compare the learned policy results with best possible equilibria. We prove properties of equilibria on parallel roads and provide a polynomial-time optimization for computing the most efficient equilibrium. Moreover, we show that in the absence of these policies, high demand and network perturbations would result in large congestion, whereas using the policy greatly decreases the travel times by minimizing the congestion. To the best of our knowledge, this is the first work that employs deep reinforcement learning to reduce congestion by indirectly influencing humans' routing decisions in mixed-autonomy traffic.**

***Keywords*—Dynamic routing, reinforcement learning, mixed-autonomy traffic**

I. Introduction

CONGESTION can result in substantial economic and social costs [**schrank2015**] which have only been growing in recent years, especially with the advent of ride-hailing services [**henao2017impacts**, **beojone2021inefficiency**]. Congestion is formed by a number of mechanisms, such as when many vehicles try to enter a road at the same time. A higher-level cause is from how people choose their routes – when people selfishly choose the quickest routes available to them, this often results in greater congestion and longer travel time than if people had their routes chosen for them optimally in terms of the overall experienced delay [**roughgarden2002bad**]. There are some existing methods for fighting congestion, such as congestion pricing [**hu2019newYork**], variable speed limits [**lu2011novel**] and highway ramp metering [**gomes2008behavior**]. However, they can be difficult to administer, and can require significant changes to infrastructure.

The introduction of autonomous vehicles to public roads provides an opportunity for better congestion management [**di2020survey**]. Our key idea is that by controlling the routing of autonomous vehicles, we can change the delay associated with traversing each road, thereby indirectly influencing peoples' routing choices. By influencing people to use more "socially advantageous" routes, we can eliminate long queues and significantly reduce traffic jams on roads.

The model for *mixed-autonomy* traffic, meaning traffic with both human-driven and autonomous vehicles, is complex, involving very large and continuous state space and continuous action space. Having human drivers dynamically respond to the choices of the autonomous vehicles further complicates the matter, making a dynamic programming-based approach and other classical methods infeasible. Because of this, we use model-free deep reinforcement learning (RL) to learn a policy without requiring access to the dynamics of the transportation network. Specifically, we show it is possible to learn a policy via proximal policy optimization (PPO) [**schulman2017proximal**] that mitigates traffic congestion by managing routing of autonomous cars given the network state.

To understand the performance of the learned policy, we investigate the *equilibrium* behavior of the network. Previous works [**krichene2017stackelberg**, **lazar2018altruism**] have shown that there is a wide spectrum of equilibria in traffic networks, meaning situations in which everyone is taking the quickest route immediately available to them, and these equilibria can have greatly varying average user delay. We establish efficient ways to compute equilibria in the network and compare the best equilibrium (in terms of latency) with the RL policy, which works regardless of whether equilibrium conditions hold or not. We show that the learned policy reaches the 'desirable' equilibria that have low travel times when starting with varying traffic patterns, and can recover network functionality after a disturbance such as a traffic accident. To summarize, our contributions are as follows:

- *Theoretical analysis:* We characterize equilibria in the network and derive a polynomial-time computation for finding optimal equilibria of parallel networks.
- *Finding a control policy via deep RL:* We employ deep RL methods to learn a routing policy for autonomous cars that effectively saves the traffic network from unboundedly large delays. We show via simulation that the RL policy is able to bring our network to the best possible equilibrium when starting from a congested state or after a network disturbance on parallel networks. We further show that an MPC-based approach and a greedy optimization method fail to do so, and thus is outperformed by the RL-based method in general networks.

We visualize our framework in the schematic diagram Fig. 1.

Literature review. Many works seek to understand how much traffic network latency could be improved if vehicle routing was controlled by a central planner, including works on *congestion games* [**dafermos1972multiclass_user**, **hearn1984convex**, **roughgarden2002bad**, **lazar2020routing**,

*Authors contributed equally.

Daniel Lazar is with the Department of Electrical and Computer Engineering, UC Santa Barbara dlazar@ece.ucsb.edu

Erdem Bıyık is with the Department of Electrical Engineering, Stanford University ebiyik@stanford.edu

Dorsa Sadigh is with the Departments of Computer Science and Electrical Engineering, Stanford University dorsa@cs.stanford.edu

Ramtin Pedarsani is with the Department of Electrical and Computer Engineering, UC Santa Barbara ramtin@ece.ucsb.edu

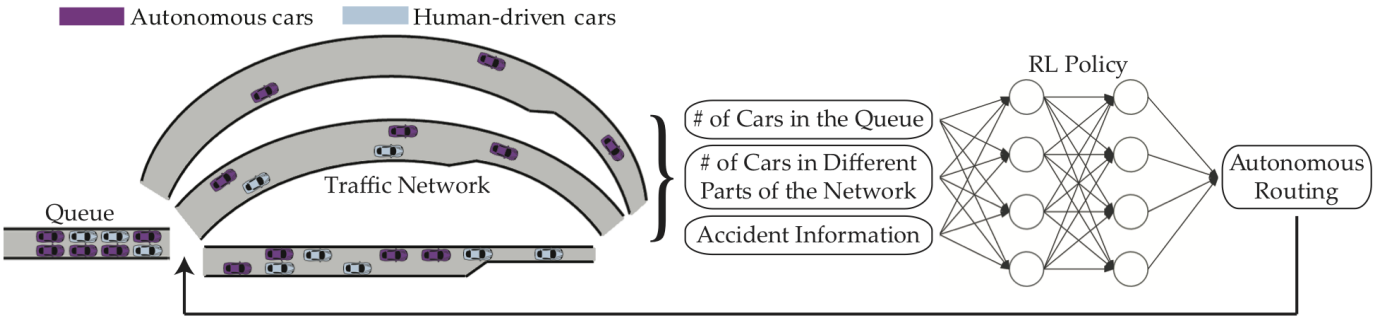

Fig. 1: The schematic diagram of our framework. Our deep RL agent processes the state of the traffic and outputs a control policy for autonomous cars' routing.

mehr2018can]. Some study how indirectly influencing peoples' routing choices by providing them network state information affects network performance [**lazarus2018decision**, **wu2018value**]. *Stackelberg Routing*, in which only some of the vehicles are controlled, is another way to influence routing [**roughgarden2004stackelberg**, **swamy2012effectiveness**]; some works incorporate the *dynamics* of human routing choices [**krichene2018social**]. While providing useful techniques for analysis, the congestion game framework does not reflect a fundamental empirical understanding about vehicle flow on roads, namely that roads with low vehicle density have a roughly constant latency, and roads with high density see latency *increase* as flow *decreases*.

Works on CTM [**daganzo1994cell**, **muralidharan2009freeway**] capture this phenomenon, including works that characterize equilibria on roads described with CTM [**gomes2008behavior**]. Notably, some consider equilibria of parallel-path Stackelberg Games, including with mixed autonomy [**krichene2017stackelberg**, **lazar2018altruism**]. However, their analyses are limited to steady-state and do not capture the dynamics. [**aswani2011game**] considers a Fundamental Diagram of Traffic-based model for slowly varying traffic. They formulate this as a Stackelberg Game and design routing information for users to minimize overall latency and bound the resulting inefficiency in a simple network. However, they only consider a single-vehicle type, not a mixed-autonomy setting.

Some works look at the low-level control of autonomous cars, specifically controlling acceleration to smooth flow and ease congestion at bottlenecks [**cui2017stabilizing**, **wu2017emergent**, **wu2018stabilizing**]; [**vinitsky2018benchmarks**] provides a benchmark for gauging the performance of these techniques. Other works learn ramp metering policies [**belletti2018expert**], localize congestion [**sivaranjani2015localization**], and model lane-change behavior with a neural network [**wright2019neural**].

In addition to these learning methods, there has also been an effort to use RL for route selection [**mao2018reinforcement**] and driver choice modeling in traffic assignment problem [**bazzan2016multiagent**, **zhou2020reinforcement**, **ramos2018analysing**, **stefanello2016using**]. Again using RL, [**grunitzki2014individual**] shows reward shaping mechanisms could be utilized to reach better equilibria. Recently, [**shou2020reward**, **shou2020multi**] develop a hierarchical approach to optimize fares, tolling and signal control in the high-level whereas a multi-agent RL method models the drivers in the lower level. Although these works show the effectiveness and potential of RL methods in transportation, to the best of our knowledge, these methods have not been used in a routing game with mixed-autonomy traffic where a central planner aims to reduce congestion by indirectly influencing humans' routing via the routing of autonomous vehicles.

Without any reinforcement learning component, some works provide macroscopic models of roads shared between human-driven and autonomous cars. [**li2018modeling**] models highway bottlenecks in the presence of platoons of autonomous vehicles mixed in with human-driven vehicles. The authors relate their model to a CTM type-model similar to the model presented below, though it is specific to a single highway. [**mahmassani201650th**] describes a microscopic model to determine the effect of autonomy on throughput, yielding fundamental diagrams. The fundamental relationship between autonomy level and critical density in our model mirrors that of [**levin2016multiclass**], which develops a CTM model for mixed autonomy traffic.

Some works solve the dynamic traffic assignment problem for networks with a CTM-based flow model, including some which decompose the optimization to enable optimizing flow on large networks [**mehrabipour2019decomposition**]. In contrast, our works studies the setting in which some flow demand is controlled to optimize the system performance, and some flow demand updates according to a selfish update rule. This precludes the use of such decomposition techniques, since the optimization can no longer be formulated as a linear program. Because of this, we use RL to solve for a routing policy in our setting.

II. Vehicle Flow Dynamics: Modeling Roads

In this section we describe dynamics governing how vehicle flow travels on a road. We extend the CTM, a widely used model that discretizes roads into *cells*, each with uniform density [**daganzo1994cell**, **muralidharan2009freeway**], for mixed-autonomy traffic. In CTM, each road segment has a maximum flow that can traverse it. The key idea of our extension is that since autonomous vehicles can keep a shorter headway (distance to the car in front of it), the greater the fraction of autonomous vehicles on a road, the greater the maximum flow that the road can serve [**lazar2018altruism**]. Accordingly, our extension of CTM lies in the dependence of cell parameters on the *autonomy level*, or the fraction of autonomous vehicles, in each cell.

We use our capacity model in conjunction with Daganzo's CTM formulation in [**daganzo1994cell**, **daganzo1995cell**], the combination of which we describe in the following. We consider a network of roads with a single origin and destination for all vehicles in the network. The origin and destination are connected by the set of simple paths $\mathcal{P}$. Each path is composed of a number of cells, and we denote the set of cells composing path p by $\mathcal{I}_p$. We generally use i and p as indices for cells and paths, respectively.

In the CTM, every cell has a *critical density*, and when the density of a cell exceeds the critical density, that cell is *congested*. We model the critical density as being dependent on the autonomy level. This is because autonomous vehicles maintain a different nominal headway than human-driven vehicles; in other words, autonomous vehicles may require more space in front of them due to prediction error, or less space, as they may react faster than human drivers. Accordingly, we use the model in [**lazar2017routing**] to model the capacity of a cell.

Using this model, each cell i has a free-flow velocity, $\bar{v}_i$, as well as a nominal headway for vehicles traveling at the free-flow velocity — h_i^{h} cells/vehicle for human-driven vehicles and h_i^{a} for autonomous vehicles. The capacity of the cell then varies with the autonomy level, denoted $\alpha_i \in [0, 1]$. We use b_i to denote the number of lanes in a cell. We model vehicles as slowing down when the headway experienced decreases below the nominal headway required and accordingly model the critical density as follows, as in [**askari2017effect**, **lazar2017routing**, **lazar2018altruism**, **mehr2018can**]:

$$\tilde{n}_i(\alpha_i) := b_i/(\alpha_i h_i^{\mathrm{a}} + (1-\alpha_i)h_i^{\mathrm{h}}). \tag{1}$$

Each cell also has a vehicle density, $n_i = n_i^{\mathrm{h}} + n_i^{\mathrm{a}}$, where n_i^{h} and n_i^{a} are, respectively, the number of human-driven and autonomous vehicles. Thus, $\alpha_i = n_i^{\mathrm{a}}/(n_i^{\mathrm{h}} + n_i^{\mathrm{a}})$. As the cells are very large compared to the vehicles, we consider these quantities to be continuous variables. As mentioned above, CTM has two regimes for vehicle flow: free-flow, when cell density is less than the critical density, and congestion, when cell density is greater than the critical density but less than the *jam density* $\bar{n}_i$, the density at which flow stops completely.

Three factors limit the flow from one cell to another. One is the *capacity*, or maximum flow out of a cell, which is the flow of vehicles that traverse the cell at the critical density:

$$\bar{F}_i(\alpha_i) := \bar{v}_i \tilde{n}_i(\alpha_i)\ . \tag{2}$$

The flow out of a cell is limited by the sending function of that cell, which is the minimum of the capacity of the cell and the *demand* of vehicles in the cell: $S_i(\alpha_i(k)) = \min(\bar{F}_i(\alpha_i), \bar{v}_i n_i(k))$. The flow entering a cell is limited by that cell's receiving function, which is the minimum of its capacity and its *supply* of vehicles: $R_i(\alpha_i(k)) = \min(\bar{F}_i(\alpha_i), (\bar{n}_i - n_i)w_i(\alpha_i))$, where w_i is the *shockwave speed*, the speed at which slowing waves of traffic propagate upstream: $w_i(\alpha_i) := \bar{v}_i\tilde{n}_i(\alpha_i)/(\bar{n}_i - \tilde{n}_i(\alpha_i))$. In the following, we use $f_i(k)$ to denote the flow out of cell i at time k and $y_i(k)$ to denote the flow into cell i. We use the standard superscripts for human-driven and autonomous flow, with the relationships $f_i^{\mathrm{h}}(k) + f_i^{\mathrm{a}}(k) = f_i(k)$ and $y_i^{\mathrm{h}}(k) + y_i^{\mathrm{a}}(k) = y_i(k)$. Accordingly,

$$\begin{aligned} n_i^{\mathrm{h}}(k+1) &= n_i^{\mathrm{h}}(k) + y_i^{\mathrm{h}}(k) - f_i^{\mathrm{h}}(k)\ , \\ n_i^{\mathrm{a}}(k+1) &= n_i^{\mathrm{a}}(k) + y_i^{\mathrm{a}}(k) - f_i^{\mathrm{h}}(k)\ . \end{aligned} \tag{3}$$

Since some cells might be a part of more than one path, we also track the paths of the human-driven and autonomous vehicles in each cell. We use $\mu_i^{\mathrm{h}}(p,k)$ and $\mu_i^{\mathrm{a}}(p,k)$ to denote the fraction of human-driven and autonomous vehicles, respectively, in cell i at time k that are taking path p. If cell i is not on path p, let $\mu_i^{\mathrm{h}}(p,k) = \mu_i^{\mathrm{a}}(p,k) = 0$.

Extending the development in [**blubook_vol1_v085**], we formulate a calculation of the flow of mixed autonomous vehicles through general junctions. We define O as the set of intersections, or junctions, in the network. We use $\Xi(o)$ to denote the set of turning movements through intersection o, with a turning movement denoted by a tuple, such as $[i, o, j] \in \Xi(o)$, where i denotes the incoming cell, and j denotes the outgoing cell. As before, we consider all cells to have one direction of travel. For intersection o we define a set of conflict points $C(o)$, and $\Xi(c)$ denotes the set of turning movements through the intersection which pass through conflict point c, where $c \in C(o)$. These routes may have different priority levels, so for each $[i, o, j] \in \Xi(c)$ we define $\beta_{ioj}^c > 0$ as the priority of turning movement $[i, o, j]$ through conflict point c. Each conflict point has some supply R_c, which we assume is independent of the level of autonomy of the vehicles passing through it. The relative priority of the turning movements will determine the relative flow of each turning movement through the conflict point. In a slight abuse of notation, we use $f_{ioj}(k)$ to denote the *total* flow of vehicles through turning movement $[i, o, j]$ at time k; we use $f_{ioj}^{\mathrm{h}}(k)$ and $f_{ioj}^{\mathrm{a}}(k)$ to denote the flow of human and autonomous vehicles, respectively, through the turning movement. We use $\Gamma(o)$ and $\Gamma^{-1}(o)$ to denote the set of cells exiting and entering junction o, respectively.

We then calculate the flows at each time step as in Algorithm 1.

An interpretation of this algorithm is as follows. The set A denotes the set of turning movements with flows that can yet be increased, and each turning movement is assigned a rate at which its flow increases. As sending and receiving limits are reached, turning movements are removed from A until there are no more turning movements left to increase.

In more concrete terms, first calculate the fraction of vehicles in each incoming cell which are headed to each outgoing cell. Then initialize all flows to 0 and initialized the unused sending and receiving capacity for each cell and conflict point. We then find relative rates of flow increase, δ_{ioj}, for the turning movements. In the loop, we calculate the similar rates of flow increases for the receiving cells and conflict points based on the rates previously found. Then, the flows are increased by the established rates until either a sending limit, a cell receiving limit, or conflict point capacity is reached. Any turning movement that has reached its sending limit is removed from the set of turning movements with further flow increases, A. Similarly, any turning movement that exits from a cell which has reached its receiving limit is removed from A, and the same with turning movements through conflict points which have reached their capacity. The loop repeats until A is empty.

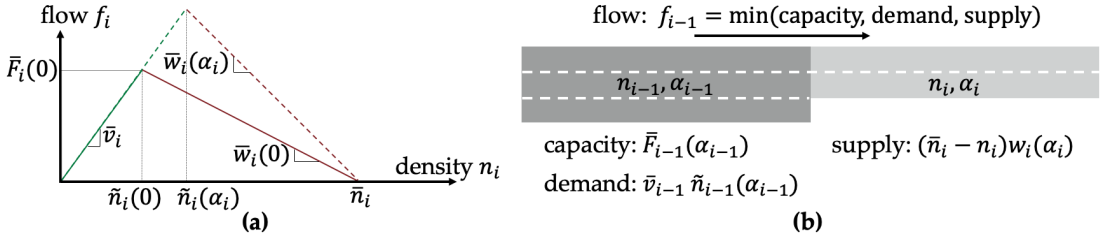

Fig. 2: **(a)** Fundamental diagram of traffic governing vehicle flow in each cell of the Cell Transmission Model. The solid line corresponds to a cell with only human-driven vehicles; the dashed line represents a cell with both vehicle types at autonomy level α_i. Green and red respectively represent a cell in free-flow and congestion. **(b)** The flow from one cell to another is a function of the density n and autonomy level α in each cell. In both figures, we suppress the notation for path p.

Having calculated the flow through the intersection, the states of each cell is updated as follows. We compute the incoming flows for the outgoing cells as follows:

$$\begin{aligned} \forall (o,j) \in \Gamma(o),\, y_j^{\mathrm{h}}(k) &= \sum_{[i,o,j]\in\Xi(o)} f_{ioj}^{\mathrm{h}} \\ y_j^{\mathrm{a}}(k) &= \sum_{[i,o,j]\in\Xi(o)} f_{ioj}^{\mathrm{a}} \\ y_j(k) &= y_j^{\mathrm{h}}(k) + y_j^{\mathrm{a}}(k) \end{aligned} \tag{4}$$

To calculate the outgoing flows of the incoming cells,

$$\begin{aligned} \forall (i,o) \in \Gamma^{-1}(o),\, f_i^{\mathrm{h}}(k) &= \sum_{[i,o,j]\in\Xi(o)} f_{ioj}^{\mathrm{h}} \\ f_i^{\mathrm{a}}(k) &= \sum_{[i,o,j]\in\Xi(o)} f_{ioj}^{\mathrm{a}} \\ f_i(k) &= f_i^{\mathrm{h}}(k) + f_i^{\mathrm{a}}(k)\,, \end{aligned} \tag{5}$$

where $\Gamma^{-1}(o)$ denotes the set of cells going into intersection o. (3) updates the human-driven and autonomous vehicle densities of each cell at the next time step. To update the fraction of vehicles in the outgoing cells on each path,

$$\begin{aligned} &\forall (o,j) \in \Gamma(o), \\ &\mu_j^{\mathrm{h}}(p,k+1) = \\ &\frac{\sum_{[i',o,j]\in\Xi(o)} f_i^{\mathrm{h}}(k)\mu_i^{\mathrm{h}}(p,k) + \mu_j^{\mathrm{h}}(p,k)(n_j^{\mathrm{h}}(k) - f_j^{\mathrm{h}}(k))}{n_j^{\mathrm{h}}(k+1)}\,, \\ &\mu_j^{\mathrm{a}}(p,k+1) = \\ &\frac{\sum_{[i',o,j]\in\Xi(o)} f_i^{\mathrm{a}}(k)\mu_i^{\mathrm{a}}(p,k) + \mu_j^{\mathrm{a}}(p,k)(n_j^{\mathrm{a}}(k) - f_j^{\mathrm{a}}(k))}{n_j^{\mathrm{a}}(k+1)}\,. \end{aligned}$$

Accidents. To evaluate the performance of the developed RL policy in reacting to disturbances, we consider stochastic accidents occuring in the network, each of which causes one lane to be closed. We let accidents occur in any cell at any time with equal probability as long as the jam density does not decrease below the current density of the cell. Each accident is cleared out after some number of time steps, drawn from a Poisson distribution. If $\bar{b}_i$ lanes of cell i are closed due to accidents, then the jam density and the critical density for the cell reduce to $(b_i - \bar{b}_i)/b_i$ of their original values. Thus, accidents introduce time-dependency to these variables.

III. Network Dynamics: Routing for Humans and Autonomous Vehicles

As mentioned above, we consider a network with a set of possible paths $\mathcal{P}$. We use λ^{h} and λ^{a} to denote the human-driven and autonomous vehicle demands, respectively. We model all vehicles entering the network as entering a *queue*, a single cell with infinite capacity. We use 0 for the index of this cell. The routing choices of autonomous vehicles leaving the queue is determined by the central controller, and the routing choices of human-driven vehicles leaving it are determined from the latencies associated with each path, detailed below.

A. Human choice dynamics

In general, people wish to minimize the amount of time spent traveling. However, people do not change routing choices instantaneously in response to new information; rather they have some inertia and only change strategies sporadically. Moreover, we assume people only account for current conditions and do not strategize based on predictions of the future [**sandholm2010population**]. Accordingly, we use an *evolutionary dynamic* to describe how a population of users choose their routes.[1] Specifically, we model the human driver population as following Hedge Dynamics, also called Log-linear Learning [**cesa2006prediction**, **marden2012revisiting**, **blume1993statistical**].

Let $(\mu_0^{\mathrm{h}}(p,k))_{p\in\mathcal{P}}$ represent the initial routing of human-driven vehicles at time k; accordingly, $\sum_{p\in\mathcal{P}} \mu_0^{\mathrm{h}}(p,k) = 1$ for all k. Humans will update their routes based on their estimates of how long it will take to traverse each path. However, it is not always possible to predict travel time accurately on general networks, since vehicles entering later on a different path may influence the travel time of vehicles entering earlier. Because of this, we consider that humans have an estimate $\hat{\ell}_p(k)$ of the true latency $\ell_p(k)$. With these estimates, the routing vector is updated as follows.

$$\mu_0^{\mathrm{h}}(p,k+1) = \frac{\mu_0^{\mathrm{h}}(p,k)\exp(-\eta^{\mathrm{h}}(k)\hat{\ell}_p(k))}{\sum_{p'\in\mathcal{P}} \mu_0^{\mathrm{h}}(p',k)\exp(-\eta^{\mathrm{h}}(k)\hat{\ell}_{p'}(k))}\,. \tag{6}$$

The ratio of the volume of vehicles using a path at successive time steps is inversely proportional to the exponential of the delay experienced by users of that path. The learning rate $\eta^{\mathrm{h}}(k)$ may be decreasing or constant. Krichene *et al.* introduce

[1] Alternately, one could model individual users as learning agents, posing it as a Multi-Agent Reinforcement Learning problem. However, we consider large networks with too many human agents for this to be feasible.

Algorithm 1 Flow Calculation

procedure FLOW CALCULATION(Intersection o)

$$\forall [i,o,j] \in \Xi(o),\ p^{\mathrm{h}}_{ioj} \leftarrow \sum_{p\in\mathcal{P}: i,j\in\mathcal{I}_p} \mu^{\mathrm{h}}_i(p,k)$$

$$p^{\mathrm{a}}_{ioj} \leftarrow \sum_{p\in\mathcal{P}: i,j\in\mathcal{I}_p} \mu^{\mathrm{a}}_i(p,k)$$

$$p_{ioj} \leftarrow \frac{n^{\mathrm{h}}_i(k)p^{\mathrm{h}}_{ioj} + n^{\mathrm{a}}_i(k)p^{\mathrm{a}}_{ioj}}{n^{\mathrm{h}}_i(k) + n^{\mathrm{a}}_i(k)}$$

$$\forall [i,o,j] \in \Xi(o),\ f_{ioj} \leftarrow 0$$

$$f^{\mathrm{h}}_{ioj} \leftarrow 0$$

$$f^{\mathrm{a}}_{ioj} \leftarrow 0$$

$$\tilde{S}_{ioj} \leftarrow S_i(\alpha_i(k))p_{ioj}$$

$$\forall (o,j) \in \Gamma(o),\ \tilde{R}_{oj} \leftarrow R_j(\alpha_j(k))$$

$$\forall c \in C(o),\ \tilde{R}_c \leftarrow R_c$$

For all $[i,o,j] \in \Xi(o)$, set δ_{ioj} such that

$$\forall [i,o,j] \in \Xi(o),$$

$$\forall [i',o,j'] \in \Xi(o),\ \frac{\delta_{ioj}}{\delta_{i'oj'}} = \frac{p_{ioj}}{p_{i'oj'}}\ ,$$

where i can equal i' and j can equal j', and

$$\forall c \in C(o);\ \forall [i,o,j] \in \Xi(c)$$

$$\forall [i',o,j'] \in \Xi(c),\ \frac{\delta_{ioj}}{\delta_{i'oj'}} = \frac{\beta^c_{ioj}p_{ioj}}{\beta^c_{i'oj'}p_{i'oj'}}\ ,$$

where i can equal i' and j can equal j'.

$A \leftarrow \Xi(o)$

while $A \neq \emptyset$ **do**

$$\forall (o,j) \in \Gamma(o),\ \delta_{oj} \leftarrow \sum_{[i,o,j]\in A} p_{ioj}$$

$$\forall c \in C(o),\ \delta_c \leftarrow \sum_{[i,o,j]\in \Xi(c)\cap A} p_{ioj}$$

$$\theta = \min\{ \min_{[i,o,j]\in A} \frac{\tilde{S}_{ioj}}{\delta_{ioj}}, \min_{(o,j)\in\Gamma(o),\delta_{oj}>0} \frac{\tilde{R}_{oj}}{\delta_{oj}}, \min_{c\in C(o):\delta_c>0} \frac{\tilde{R}_c}{\delta_c}\}$$

$$\forall [i,o,j] \in A,\ f_{ioj} \leftarrow f_{ioj} + \theta\delta_{ioj}$$

$$f^{\mathrm{h}}_{ioj} \leftarrow f^{\mathrm{h}}_{ioj} + \theta\delta_{ioj}(1-\alpha_i(k))$$

$$f^{\mathrm{a}}_{ioj} \leftarrow f^{\mathrm{a}}_{ioj} + \theta\delta_{ioj}\alpha_i(k)$$

$$\tilde{S}_{ioj} \leftarrow \tilde{S}_{ioj} - \theta\delta_{ioj}$$

$$\forall (oj) \in \Gamma(o),\ \tilde{R}_{oj} \leftarrow \tilde{R}_{oj} - \theta\delta_{oj}$$

$$\forall c \in C(o),\ \tilde{R}_c \leftarrow \tilde{R}_c - \theta\delta_{oj}$$

$$A \leftarrow A \setminus \{[i,o,j] \in A : \tilde{S}_{ioj} = 0\}$$

$$A \leftarrow A \setminus \{[i,o,j'] \in A : \tilde{R}_{oj} = 0 \wedge p_{ioj'} > 0\}$$

$$A \leftarrow A \setminus \{[i,o,j] \in c : \tilde{R}_c = 0\ \forall c \in C(o)\}$$

end while

return f_{ioj}, f^{h}_{ioj}, f^{a}_{ioj}, $\forall [i,o,j] \in \Xi(o)$

end procedure

this model in the context of humans' routing choices and simulate a congestion game with Amazon Mechanical Turk users to show the model accurately predicts human behavior [**krichene2018learning**]. We note that though we use this specific model for human choice for our simulations, the control method described later does not require this specific choice of human choice model. Our theoretical analysis similarly is not restricted to this choice of dynamics and works for any human choice model in which all fixed points of the dynamics satisfy human selfishness.

B. Autonomous vehicle control policy

We assume that we have control over the routing of autonomous vehicles. We justify this by envisioning a future in which autonomous vehicles are offered as a service rather than a consumer product. We then assume that a city can coordinate with the owner of an autonomous fleet to decrease congestion in the city. Moreover, unlike traditional tolling, coordination between autonomous vehicles and city infrastructure allows for fast-changing and geographically finely quantized tolls, enabling routing control to be achieved through incentives [**biyik2019green**, **beliaev2021incentivizing**]. The initial routing of autonomous vehicles is then our control parameter by which we influence the state of traffic on the network. Consistent with the previous notation, we denote the initial autonomous routing as $(\mu^{\mathrm{a}}_0(p,k))_{p\in\mathcal{P}} \in \mathbb{R}^{|\mathcal{P}|}_{\geq 0}$, where $\sum_{p\in\mathcal{P}} \mu^{\mathrm{a}}_0(p,k) = 1$.

We assume the existence of a central controller, or social planner, which dictates μ^{a}_0 by processing the state of the network. At each time step, we let the controller observe:

- the number of human-driven and autonomous vehicles in each cell and in the queue,
- binary states for each lane that indicates whether the lane is closed due to an accident or not.

We use deep RL to arrive at a policy for the social planner to control the autonomous vehicle routing, μ^{a}_0. Since the state space is very large and both state and action spaces are continuous, a dynamic programming-based approach is infeasible. For instance, even if we discretized the spaces, say with 10 quantization levels, and did not have accidents, we would have 10^{82} possible states and 10 actions for a moderate-size network with only 2 paths and 40 cells in total.

We wish to minimize the total latency experienced by users, which is equal to summing over time the number of users in the system at each time step. Accordingly, the stage cost is:

$$J(k) = \sum_{i\in\mathcal{I}} n_i(k)\ . \tag{7}$$

Due to their high performance in continuous control tasks [**schulman2015trust**, **schulman2017proximal**], we employ policy gradient methods to learn a policy that produces μ^{a}_0 given the observations. Specifically, we use state-of-the-art PPO with an objective function augmented by adding an entropy bonus for sufficient exploration [**schulman2017proximal**, **mnih2016asynchronous**]. We build a deep neural network, and train it using Adam optimizer [**kingma2014adam**]. An overview of the PPO method and the set of parameters we use are presented in the appendix (Sec. VII-C and Sec. VII-D). Each episode has a fixed number of time steps.

In order to evaluate the performance of our control policy, we use three criteria. The first is the *throughput* of the network – we wish to have a policy that can serve any feasible demand, thereby stabilizing the queue. The second is the *average delay* experienced by users of the network, which we measure by counting the number of vehicles in the system. The third is the *convergence* to some steady state; we wish to avoid wild oscillations in congestion. To contextualize the performance of our control policy in this framework, we first establish the performance of *equilibria* of the network.

IV. Equilibrium analysis

In this section, we examine the possible *equilibria* of our dynamical system, which characterize the possible steady state behaviors of the system. A network with a given demand can have a variety of equilibria with varying average user delay. If our control achieves overall delay equal to that of the best possible equilibrium, it is a successful policy. Section V shows empirically that our learned policy can achieve the best equilibrium in a variety of settings.

In this section, we first formulate an optimization which solves for the most efficient equilibrium, which is computationally hard. Motivated by this, we restrict the class of networks considered and prove theoretical properties of this restricted class. Using these properties, we formulate a new optimization formulation to solve for the most efficient equilibrium and prove it is solvable in polynomial time.

A. Equilibrium Formulation

We define two notions of equilibrium: one related to the vehicle flow dynamics, and one related to human choice dynamics.[2]

Definition 1 (Path Equilibrium)**.** *We define a* path equilibrium *for path p as a set of cell densities $(n_i^h(k), n_i^a(k))_{i\in\mathcal{I}_p}$ that, for a given constant flow entering the first cell on the path, $y_i^h(k)$ and $y_i^a(k)$, the cell densities are constant.*

Definition 2 (Network Equilibrium)**.** *We define a* network equilibrium *as a set of cell densities $(n_i^h(k), n_i^a(k))_{i\in\mathcal{I}}$ and human vehicle routing $(\mu_0^h(p,k))_{p\in\mathcal{P}}$, such that for a given constant entering flow $y_0^h(k)$ and $y_0^a(k)$ and a given constant autonomous vehicle routing $(\mu_0^a(p,k))_{p\in\mathcal{P}}$, the human vehicle routing, subject to the dynamics in* (6)*, is constant.*

We are interested in satisfying both notions of equilibrium – both the path equilibrium, which deals with the vehicle flow dynamics, and the network equilibrium, which deals with the human choice dynamics. Accordingly, the pair can be considered a Stackelberg Equilibrium for a leader controlling the autonomous vehicles who wishes to maximize the social utility in the presence of selfish human demand. We formulate the following optimization to solve for the most efficient equilibrium (satisfying both notions of equilibrium defined above), *i.e.* the equilibrium which minimizes the total travel time of all users of the network. We drop all time indices since we consider quantities that are constant over time.

$$\min_{(n_i^{\mathrm{h}},n_i^{\mathrm{a}},f_i^{\mathrm{h}},f_i^{\mathrm{a}},y_i^{\mathrm{h}},y_i^{\mathrm{a}},\mu_i^{\mathrm{h}}(p),\mu_i^{\mathrm{a}}(p),\ell_p)_{i\in\mathcal{I},p\in\mathcal{P}}} \sum_{i\in\mathcal{I}} n_i$$

s.t. $\forall o \in O$: **procedure** Flow Calculation(Intersection o)

$$(4), (5)$$

$$\forall i \in \mathcal{I} : y_i^{\mathrm{h}} = f_i^{\mathrm{h}}, \quad y_i^{\mathrm{a}} = f_i^{\mathrm{a}}, \quad \alpha_i = n_i^{\mathrm{a}}/(n_i^{\mathrm{h}} + n_i^{\mathrm{a}})$$

$$\sum_{i'\in\mathcal{U}_i} f_{i'}^{\mathrm{h}}\mu_{i'}^{\mathrm{h}}(p) = f_i^{\mathrm{h}}\mu_i^{\mathrm{h}}(p), \ \sum_{i'\in\mathcal{U}_i} f_{i'}^{\mathrm{a}}\mu_{i'}^{\mathrm{a}}(p) = f_i^{\mathrm{a}}\mu_i^{\mathrm{a}}(p)$$

$$\ell_p = \sum_{i\in p} (n_i^{\mathrm{h}} + n_i^{\mathrm{a}})/(f_i^{\mathrm{h}} + f_i^{\mathrm{a}})$$

$$\forall p, p' \in \mathcal{P} : \mu_0^{\mathrm{h}}(p)(\ell_p - \ell_{p'}) \le 0$$

While this formulation solves for the most efficient equilibrium of any traffic network, it is computationally difficult, especially due to the final constraint. Due to this, we introduce a restricted class of networks that we consider for the remainder of this section, which allows us to compute equilibria in polynomial time with respect to the number of paths.

Definition 3 (Bottleneck)**.** *We define a* bottleneck *as a regular junction at which the number of lanes decreases, decreasing the capacity of the cells.*

Assumption 1. *We consider a parallel network in which leaving the first cell, vehicles choose a path and paths do not share cells, meaning that each cell is identified with only one path, aside from the downstream-most cell which has infinite capacity. We further consider that all cells in the path have the same model parameters, except for a bottleneck after the m_p^n upstream-most cells.*

In other words, we consider a parallel network where each path is composed of identical cells except for a single junction with a decrease in the number of lanes. Fig. 1 shows an example of such a network. For ease of analysis, we first establish properties of Path Equilibria, then Network Equilibria.

B. Path equilibrium

As mentioned above, we restrict our considered class of paths to those with a single bottleneck, meaning one point on the path at which cell capacity drops. Formally, we consider each path p to have m_p^{n} cells, each with b_p^{n} lanes, followed by m_p^{b} cells downstream, each with b_p^{b} lanes, where $b_p^{\mathrm{b}} < b_p^{\mathrm{n}}$. We define $r_p := b_p^{\mathrm{b}}/b_p^{\mathrm{n}} \in (0,1)$.

In a slight abuse of notation, we use the subscript p for parameters that are constant over a path under Assumption 1, and the superscript n for cells before the bottleneck and b for the bottleneck and cells downstream of it. We now present a theoretical result that completely analytically characterizes the path latencies that can occur at equilibrium.

Theorem 1. *Under Assumption 1 a path p with flow dynamics described in Section II that is at Path Equilibrium will have the same autonomy level in all cells. Denote this autonomy level α_p. If the vehicle flow demand is strictly less than the minimum cell capacity, the path will have no congested cells.*

[2]These define equilibria in the sense of dynamical systems, and do not strictly correspond to game-theoretic notions of equilibria. Under Assumption 3 below, the set of equilibria for the dynamics of human choice will correspond to the set of Nash Equilibria where the payoff is the path latency.

Otherwise, the path will have one of the following latencies, where $\gamma_p \in \{0, 1, 2, \ldots, m_p^n\}$:

$$\ell_p = \frac{|\mathcal{I}_p|}{\bar{v}_p} + \gamma_p \frac{(1-r_p)\bar{n}_p^n(\alpha_p h_p^a + (1-\alpha_p)h_p^h)}{r_p \bar{v}_p b_p^n} \ .$$

Proof. The proof is composed of three lemmas. We first establish a property of path equilibria that allows us to treat the vehicle flow as if it were composed of a single car type. With this, we use the CTM to characterize possible equilibria on a path. We then derive the delay associated with each congested cell. Combining the latter two lemmas yields the theorem.

Lemma 1. *A path in equilibrium with nonzero incoming flow has the same autonomy level in all cells of the path, which is equal to the autonomy level of the vehicle flow onto the path. Formally, a path p with demand $(\bar{\lambda}_p^h, \bar{\lambda}_p^a)$ in equilibrium has, for all cells i in $\mathcal{I}_p$,*

$$\alpha_i = \bar{\lambda}_p^a / (\bar{\lambda}_p^h + \bar{\lambda}_p^a) \ .$$

We defer the proof of the lemma to the appendix. With this, our path equilibria analysis simplifies to that of single-typed traffic, with the autonomy level treated as a variable parameter. The next lemma, similarly to Theorem 4.1 of [**gomes2008behavior**], completely characterizes the congestion patterns that can occur in cell equilibria. For this lemma, we consider the cell indices in a path to be increasing, where the cell immediately downstream from a cell i has index $i+1$.

Lemma 2. *Under Assumption 1, if the demand on a path is less than the minimum capacity of its cells, they will be uncongested at path equilibrium. Otherwise, a path with demand equal to the minimum cell capacity will have m_p^n possible path equilibria, corresponding to one of the following sets of congested cells, where j is the index of the m_p^nth cell:*

$$\big\{\emptyset, \{j\}, \{j-1, j\}, \ldots, \{j - m_p^n + 1, \ldots, j-2, j-1, j\}\big\} \ .$$

Proof. As mentioned above, this lemma relates closely to Theorem 4.1 of [**gomes2008behavior**]. However, we cannot directly apply that theorem due to differing assumptions; namely they assume $\bar{F}_{i+1} = (\bar{n}_i - \tilde{n}_i)w_i$ for all i. We therefore offer a similar proof, tailored to our assumptions.

For ease of notation, we drop all path subscripts p as well as the cell index for the free-flow velocity parameter $\bar{v}$. In light of Lemma 1, we also suppress the autonomy level arguments to capacity $\bar{F}_i$ and critical density $\tilde{n}_i$. The flow equation then becomes $f_i = \min(\bar{v}n_i,\ (\bar{n}_{i+1} - n_{i+1})w_{i+1},\ \bar{F}_i, \bar{F}_{i+1})$.

We begin by proving that if the vehicle flow demand is strictly less than the minimum capacity, *i.e.* the bottleneck capacity, then the only equilibrium has no congested cells. Let us use j' to denote the index of the final cell in the path. Under Assumption 1 there is no supply limit to the flow exiting a path, so $f_{j'} = \min(\bar{v}n_{j'}, \bar{F}_{j'})$. Since $f_0 = f_{j'} < \bar{F}_{j'}$, $f_0 = f_{j'} = \bar{v}n_{j'}$. The definition of capacity, $\bar{F}_i = \bar{v}\tilde{n}_i$, then implies that $n_{j'} < \tilde{n}_{j'}$, meaning that cell j' is uncongested, so $\bar{v}n_{j'} < (\bar{n}_{j'} - n_{j'})w_{j'}$.

This is the base case for a proof by induction. Consider cell i that is uncongested (*i.e.* $n_i < \tilde{n}_i$). Since by assumption all cells have flow strictly less than the cell's capacity, $f_i = \bar{v}n_i < \bar{F}_i$. Then consider the flow entering cell i: $f_{i-1} = \min(\bar{v}n_{i-1}, (\bar{n}_i - n_i)w_i, \bar{F}_{i-1}) = f_i < \bar{F}_i < (\bar{n}_i - n_i)w_i$.

The fact that $\bar{F}_i \le \bar{F}_{i-1}$ then implies that $f_{i-1} = \bar{v}n_{i-1}$, so cell $i-1$ is uncongested, proving the lemma's first statement.

The second statement assumes the flow on the path is equal to the minimum capacity. The cells in the bottleneck segment all have the same capacity, which we denote $\bar{F}^{\mathrm{b}}$; this capacity is less than the capacity of the cells in the nonbottleneck segment. This means all bottleneck cells will be operating at capacity (and therefore have vehicle density equal to their critical density); flow on the path is therefore equal to $\bar{F}^{\mathrm{b}}$.

We now turn to the nonbottleneck segment. We first note that if a nonbottleneck cell is uncongested then the preceeding cell must be uncongested as well, using the same reasoning as that proving the first statement above. Next, consider the flow out of the downstream-most cell of the nonbottleneck segment: $f_j = \min(\bar{v}n_j, (\bar{n}_{j+1} - n_{j+1})w_{j+1}, \bar{F}_j) = \bar{F}^{\mathrm{b}} < \bar{F}_j$, so $f_j = \min(\bar{v}n_j, (\bar{n}_{j+1} - n_{j+1})w_{j+1})$. Cell j can be uncongested, in which case the cell density is such that $\bar{v}n_j = \bar{F}^{\mathrm{b}}$, or the cell can be congested, in which case the second term dominates. Then, if nonbottleneck cell i is congested, the flow into it is $f_{i-1} = \min(\bar{v}n_{i-1}, (\bar{n}_i - n_i)w_i)$. Again, to achieve this flow, cell $i-1$ can be either congested or uncongested. As shown above, if uncongested, then all upstream cells must be uncongested as well, yielding the second statement in the lemma. □

We use these properties to find a closed-form expression for the latency incurred by traveling through a bottleneck cell, which when combined with Lemma 2, completes the proof.

Lemma 3. *The latency incurred by traveling through a congested cell is as follows.*

$$\frac{1}{\bar{v}_p} + \frac{(1-r_p)\bar{n}_p^n(\alpha_p h_p^a + (1-\alpha_p)h_p^h)}{r_p \bar{v}_p b_p^n} \ .$$

Proof. Recall that we assume paths have a uniform free-flow velocity across all cells in a path, where path p has free-flow velocity $\bar{v}_p$. We define $[m_p^{\mathrm{n}}]$ as the set of cells before the bottleneck, which have b_p^{n} lanes. The remaining cells, with indices in the set $\mathcal{I}_p \setminus [m_p^{\mathrm{n}}]$, have b_p^{b} lanes. Further recall the definition $r_p = b_p^{\mathrm{b}}/b_p^{\mathrm{n}}$. Let $\bar{F}_p^{\mathrm{n}}(\alpha_p)$ denote the capacity of the cells before the bottleneck of path p with autonomy level α_p and let $\bar{F}_p^{\mathrm{b}}(\alpha_p)$ be the same for the bottleneck cell. Note that $\bar{F}_p^{\mathrm{b}}(\alpha_p) = r_p \bar{F}_p^{\mathrm{n}}(\alpha_p)$. Similarly, let $w_p^{\mathrm{n}}(\alpha_p)$ and $w_p^{\mathrm{b}}(\alpha_p)$ denote the shockwave speed for prebottleneck cells and bottleneck cell, respectively, on path p with autonomy level α_p, as with jam densities $\bar{n}_p^{\mathrm{n}}$ and $\bar{n}_p^{\mathrm{b}}$ and critical densities $\tilde{n}_p^{\mathrm{n}}(\alpha_p)$ and $\tilde{n}_p^{\mathrm{b}}(\alpha_p)$.

Lemma 2 establishes all possible combinations of congested cells that a path at equilibrium can experience. We now investigate how much delay each configuration induces on the path, parameterized by the autonomy level of the path. By Lemma 2 and the definitions of r and capacity (2),

$$f_p = \bar{F}_p^{\mathrm{b}} = r_p \bar{F}_p^{\mathrm{n}}(\alpha_p) = r_p w_p^{\mathrm{n}}(\alpha_p)(\bar{n}_p^{\mathrm{n}} - \tilde{n}_p^{\mathrm{n}}(\alpha_p)) \ . \quad (8)$$

Let $n_p^{\mathrm{c}}(\alpha_p)$ denote the vehicle density in a congested cell on path p, which we know must occur upstream of the bottleneck (Lemma 2). Then, the flow entering a congested cell before

the bottleneck is $f_p = w_p^{\mathrm{n}}(\alpha_p)(\bar{n}_p^{\mathrm{n}} - n_p^{\mathrm{c}}(\alpha_p))$. Equating this with (8), we find $n_p^{\mathrm{c}}(\alpha_p) = (1-r_p)\bar{n}_p^{\mathrm{n}} + r_p\tilde{n}_p^{\mathrm{n}}(\alpha_p)$.

To use this to find the latency incurred by traveling through a congested cell, we divide the density by the flow, as follows.

$$\frac{n_p^{\mathrm{c}}(\alpha_p)}{f_p} = \frac{n_p^{\mathrm{c}}(\alpha_p)}{\bar{F}_p^{\mathrm{b}}(\alpha_p)} = \frac{(1-r_p)\bar{n}_p^{\mathrm{n}} + r_p\tilde{n}_p^{\mathrm{n}}(\alpha_p)}{r_p\bar{v}_p\tilde{n}_p^{\mathrm{n}}(\alpha_p)}$$
$$= \frac{1}{\bar{v}_p} + \frac{(1-r_p)\bar{n}_p^{\mathrm{n}}(\alpha_p h_p^{\mathrm{a}} + (1-\alpha_p)h_p^{\mathrm{h}})}{r_p\bar{v}_p b_p^{\mathrm{n}}}. \qquad \square$$

Together, the lemmas prove the theorem. $\square$

The two terms above are the free-flow delay and the per-cell latency due to congestion, respectively. Theorem 1 allows us to calculate the possible latencies of a path as a function of its autonomy level α_p. Since in a network equilibrium all used paths have the same latency, we can calculate network equilibria more efficiently than comprehensively searching over all possible routings. However, equilibria may not exist, even with a fine time discretization – in equilibrium the path latencies must be equal, but by Theorem 1, road latency is a function of the integer γ_p. To avoid this artifact, when analyzing network equilibria we consider the cells to be small enough that we can consider the continuous variable $\gamma_p \in [0, m_p^{\mathrm{n}}]$.

C. Network equilibrium

We define the best equilibrium to be the equilibrium that serves a given flow demand with minimum latency. We are now ready to establish properties of network equilibria, as well as how to compute the best equilibria. We use the following two assumptions in our analysis of network equilibrium.

Assumption 2. *No two paths have the same free-flow latency.*

Assumption 3. *The initial choice distribution has positive human-driven and autonomous vehicle flow on each path.*

Note that the Assumption 2 is not strictly necessary but is useful for easing analysis. A similar analysis could be performed in its absence. We justify Assumption 3 by noting that humans are not entirely rational and that our choice model does not capture all reasons a person may wish to choose a route, and some small fraction of people will choose routes that seem less advantageous at first glance.

Theorem 2. *Under Assumptions 1 and 2, a routing that minimizes total latency when all users (both human drivers and autonomous users) are selfish can be computed in $O(|\mathcal{P}|^3 \log |\mathcal{P}|)$ time. A routing that minimizes total latency when human drivers are selfish and autonomous users are controlled can also be computed in $O(|\mathcal{P}|^3 \log |\mathcal{P}|)$ time.*

Proof. To establish properties of network equilibria, we introduce some notation. We use $a_p = |\mathcal{I}_p|/\bar{v}_p$ to denote the free-flow latency of path p. We also use $\mathcal{P}_{\leq a_p} = \{p' \in \mathcal{P} : a_{p'} \leq a_p\}$, which denotes the set of paths with free-flow latency less than or equal to that of path p. We similarly define the expression with other comparators, *e.g.* $\mathcal{P}_{<a_p}$ or $\mathcal{P}_{>a_p}$.

This proposition follows from Definition 2 and Assumption 3. The next lemma follows, with proof deferred to the appendix.

Proposition 1. *In a network equilibrium,*

1) *All paths with selfish drivers have the same latency, and*
2) *All paths without selfish drivers have equal or greater latency.*

Lemma 4. *If the set of equilibria contains a routing with positive flow only on paths $\mathcal{P}_{\leq a_p}$, then there exists a routing in the set of equilibria in which path p is in free-flow.*

Lemma 5. *Under Assumption 2, if some users are selfish and some users are not selfish, then the best equilibrium will have the following properties:*

1) *the path with largest free-flow latency used by selfish users will be in free-flow,*
2) *all paths with lower free-flow latency will be congested,*
3) *paths with greater free-flow latency may have nonselfish users, and*
4) *paths used with larger free-flow latency that have nonselfish users on them will be at capacity, except perhaps the path with largest free-flow latency used by nonselfish users.*

Proof. Consider a network with some selfish and some nonselfish (controlled) users. Let p denote the path with the longest free-flow latency that contains selfish users. For the purpose of contradiction, let this path contain congested cells, and let this be the best equilibrium. Fix the nonselfish flow on all roads with longer free-flow latency than p. By Lemma 4, there exists an equilibrium for the selfish users in which p is in free-flow. This results in less latency for the users on path p, and no selfish user will have greater delay (Proposition 1). This contradicts the premise, proving the first property.

The second property follows directly from Proposition 1 and Assumption 2. The third property follows from the definition of nonselfish users, which can take a path with a larger latency than other available paths. The best equilibrium minimized total latency. If there was a road with nonselfish users that was not at capacity, while another path with higher latency has positive flow, this would not be the best equilibrium, since a more efficient routing would shift flow from the higher latency path to the lower latency one. This yields the final property. $\square$

Using these properties, we prove Theorem 2. We first consider the setting in which all users are selfish. We use $\ell_p^{\mathrm{c}}(\alpha_p)$ to denote the per-cell latency due to congestion, *i.e.* $\ell_p^{\mathrm{c}}(\alpha_p) = \frac{(1-r_p)\bar{n}_p^{\mathrm{n}}(\alpha_p h_p^{\mathrm{a}} + (1-\alpha_p)h_p^{\mathrm{h}})}{r_p\bar{v}_p b_p^{\mathrm{n}}}$. Lemma 5 implies that for a given demand, all equilibria in the set of most efficient equilibria for that demand have one path that is in free-flow. We can then formulate the search for a best equilibrium as an optimization. We are helped by the fact that the best equilibria will use the minimum number of feasible paths, since all users experience the same delay. Then, for each candidate free-flow path (denote with index p'), check feasibility of only using paths $\mathcal{P}_{\leq a_{p'}}$, and choose a routing that minimizes $|\mathcal{P}_{\leq a_{p'}}|$, *i.e.* the number of roads used. The reason for minimizing the number of used roads is that all users are experiencing the same latency (Proposition 1) and in the best equilibrium, the road with flow on it that has longest free-flow latency will

be in free-flow (Lemma 5). The feasibility can be checked as follows, with an optimization that utilizes Lemma 5.

$$\begin{aligned}
&\underset{(f_p^{\mathrm{h}},f_p^{\mathrm{a}})_{p\in\mathcal{P}_{\le a_{p'}}},\ \gamma\in\prod_{p\in\mathcal{P}_{<a_{p'}}}[0,m_p^{\mathrm{n}}]}{\arg\min} \quad 1 \\
&\qquad \text{s.t.} \sum_{p\in\mathcal{P}_{\le a_{p'}}} f_p^{\mathrm{h}} = \bar{\lambda}^{\mathrm{h}},\ \sum_{p\in\mathcal{P}_{\le a_{p'}}} f_p^{\mathrm{a}} = \bar{\lambda}^{\mathrm{a}} \\
&\qquad\qquad f_{p'}^{\mathrm{h}} + f_{p'}^{\mathrm{a}} \le \bar{F}_{p'}\Big(\frac{f_{p'}^{\mathrm{a}}}{f_{p'}^{\mathrm{h}} + f_{p'}^{\mathrm{a}}}\Big) \\
&\forall p \in \mathcal{P}_{<a_{p'}} : \gamma_p \ell_p^{\mathrm{c}}\Big(\frac{f_p^{\mathrm{a}}}{f_p^{\mathrm{h}} + f_p^{\mathrm{a}}}\Big) = a_{p'} - a_p \\
&\qquad\qquad f_p^{\mathrm{h}} + f_p^{\mathrm{a}} = \bar{F}_p\Big(\frac{f_p^{\mathrm{a}}}{f_p^{\mathrm{h}} + f_p^{\mathrm{a}}}\Big)
\end{aligned}$$

The last constraint yields an affine relationship between f_p^{h} and f_p^{a} for paths $\mathcal{P}_{<a_{p'}}$. Solving for f_p^{h} and plugging into the first constraint yields an affine relationship between γ_p and f_p^{a}. This way, the optimization can be converted to a linear program, and we must solve $\log|\mathcal{P}|$ linear programs to search the minimum feasible p'.

This formulation assumes that all vehicles are selfish. If instead we consider selfish human drivers and fully controlled autonomous users, we can construct a similar optimization to find the best equilibrium. For each choice of free-flow path p', we minimize the total latency of the autonomous vehicles not on free-flow paths. We then choose the routing corresponding to free-flow path p' that minimizes total latency (which may not necessarily minimize the number of paths used by human drivers). For each candidate free-flow path p' we solve the following optimization.

$$\begin{aligned}
&\underset{(f_p^{\mathrm{h}},f_p^{\mathrm{a}})_{p\in\mathcal{P}},\ \gamma\in\prod_{p\in\mathcal{P}_{<a_{p'}}}[0,m_p^{\mathrm{n}}]}{\arg\min} \quad \sum_{p\in\mathcal{P}_{>a_{p'}}} f_p^{\mathrm{a}} a_p \\
&\qquad \text{s.t.} \sum_{p\in\mathcal{P}_{\le a_{p'}}} f_p^{\mathrm{h}} = \bar{\lambda}^{\mathrm{h}},\ \sum_{p\in\mathcal{P}} f_p^{\mathrm{a}} = \bar{\lambda}^{\mathrm{a}} \\
&\qquad\qquad f_{p'}^{\mathrm{h}} + f_{p'}^{\mathrm{a}} \le \bar{F}_{p'}\Big(\frac{f_{p'}^{\mathrm{a}}}{f_{p'}^{\mathrm{h}} + f_{p'}^{\mathrm{a}}}\Big) \\
&\forall p \in \mathcal{P}_{<a_{p'}} : \gamma_p \ell_p^{\mathrm{c}}(\alpha_p) = a_{p'} - a_p \\
&\qquad\qquad f_p^{\mathrm{h}} + f_p^{\mathrm{a}} = \bar{F}_p\Big(\frac{f_p^{\mathrm{a}}}{f_p^{\mathrm{h}} + f_p^{\mathrm{a}}}\Big) \\
&\forall p \in \mathcal{P}_{>a_{p'}} : f_p^{\mathrm{a}} \le \bar{F}^{\mathrm{b}}(1)
\end{aligned}$$

This can be reformulated as a linear program by the same mechanism. Again, we solve $\log|\mathcal{P}|$ linear programs and choose the one corresponding to the minimum feasible p'. □

Using these properties to compute optimal equilibria, we establish a framework for understanding the performance of our learned control policy. If the policy can reach the best equilibrium latency starting from arbitrary path conditions we view the policy as successful. We use this baseline to evaluate our experimental results in the following section.

A question then arises: if we have computed the best possible equilibria, why do we not directly implement that control? This approach is not fruitful, since the theoretical analysis of best equilibria gives the control policy only in the steady state. In practice, the network can start in any state, including worse equilibria, from which good equilibria will not emerge when autonomous vehicles unilaterally use their routing in the best equilibrium. Besides, our equilibrium analysis is limited to parallel networks and extending it to more general networks would yield a nonconvex optimization problem. A dynamic policy which depends on the current traffic state is therefore needed to guide the network to the best equilibrium. As shown in the following section, the policy learned via deep reinforcement learning achieves this guidance and reaches the best equilibrium in a variety of settings.

V. Experiments and results

In all of the experiments[3], we adopt the following parameters. All vehicles are 4 meters long. Human drivers keep a 2 second headway distance, whereas autonomous cars can keep 1 second. Each time step corresponds to 1 minute of real-life simulation. Each episode during deep RL training covers 5 hours of real-life simulation (300 time steps). In test time, we simulate 6 hours of real-life (360 time steps) to ensure the RL policy did not learn to minimize the latency in the first 300 time steps and leave excess vehicles in the network at the end. We divide paths into the cells such that it takes 1 time step to traverse each cell in free-flow. We initialize $n_i(0) \sim \mathrm{unif}(0, 1.2\tilde{n}_i)$ for all $i \in \mathcal{I}_p$ for all $p \in \mathcal{P}$. We set the standard deviations of the zero-mean Gaussian demand noise to be $\bar{\lambda}^{\mathrm{h}}/10$ and $\bar{\lambda}^{\mathrm{a}}/10$ for human-driven and autonomous vehicles, respectively.

Our overall control scheme can be seen in Fig. 1. As the learning model, we build a two-hidden-layer neural network, with each layer having 256 nodes. We train an RL agent for each configuration that we will describe later on in simulated traffic networks that are based on the mixed-autonomy traffic model and the dynamics that we described in Sections II and III. All trainings simulate 40 million time steps.[4] Depending on whether we evaluate our RL-based approach with (or without) the accidents, we enable (or disable) accidents at the training phase. However, we note that the number of possible accident configurations in the network is far more than the expected number of accidents during all training episodes. Hence, successfully handling accidents requires good generalization performance. Similar to accidents, the demand distributions match between training and test environments.

We compare our method with two baselines: first, a *selfish* routing scheme, where all cars are selfish and use the human choice dynamics presented in Sec. III-A, and second, a model predictive control (*MPC*) based controller which can perfectly simulate the network other than the uncertainty due to accidents and noisy demand. It plans for the receding horizon of 4 minutes and re-plans after every 1 minute to minimize the number of cars in the network using a Quasi-Newton method (L-BFGS [**andrew2007scalable**]). To increase robustness against the uncertainty, it samples 12 different simulations of the network and takes the average. We note that this MPC can only be useful in small networks where some cars can enter the network and reach the destination within the MPC horizon

[3]We make the code available in the supplementary material.

[4]Other hyperparameter values we use for PPO are in the Appendix.

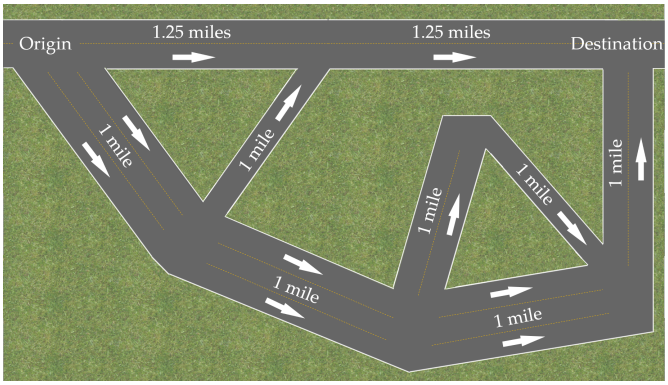

Fig. 3: The small general class network used for experiments.

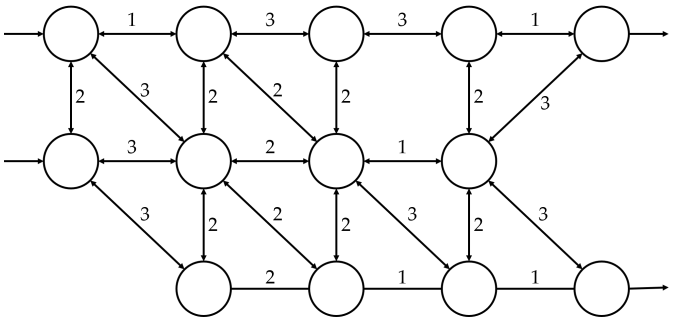

Fig. 5: OW network (adapted from [**de2011modelling**]) used for experiments.

of 4 minutes. While increasing the horizon may help MPC operate in larger networks, it causes a huge computational burden. In fact, even though we parallelized the controller over 12 Intel® Xeon® Gold 6244 CPUs (3.60 GHz), it took the controller 32 seconds on average to decide the routing of autonomous vehicles for the next 1 minute, which clearly indicates a practical problem. In all experiments, we set $\eta^{\mathrm{h}}(k)$ (and $\eta^{\mathrm{a}}(k)$ for the selfish baseline) to be 0.5 for all k.

A. General Class of Networks

We first start by considering a small network of 9 cells and 7 junctions (1 regular junction, 3 merges and 3 diverges) as shown in Fig. 3, where the priority levels of cells at merges are equal to their numbers of lanes. We set the autonomy level of the demand $\bar{\alpha}=0.6$ and the total demand $\bar{\lambda}^{\mathrm{h}}+\bar{\lambda}^{\mathrm{a}}=2.60$ cars per second. We set the probability of accidents such that the expected frequency of accidents is 1 per 100 minutes, and clearing out an accident takes 30 minutes on average [**houstontranstar2018**]. For human choice dynamics, we assume humans' latency estimates are based on the current states of each cell, i.e., they estimate the latencies as if the network is in steady-state.

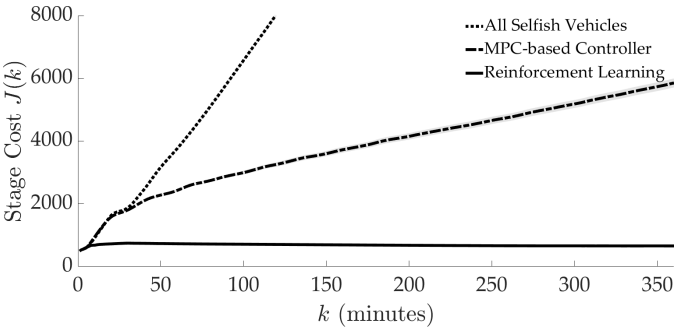

Fig. 4: Time vs. number of cars under selfish, MPC and RL routing on the small general class network.

Fig. 4 shows the number of cars in the network over time (mean $\pm$ standard error over 100 simulations). While MPC controller improves over the selfish routing, they both suffer from linearly growing queues. On the other hand, RL controller stabilizes the queue and keeps the network uncongested.

Next, we consider a larger network shown in Fig. 5 as a graph where the numbers noted on the links denote the number of cells in that link in one direction. Each cell, excluding queues which has infinite capacity, has 2 lanes. This is a quantized version of the OW network due to Ortúzar and Willumsen [**de2011modelling**], and is widely used in the transportation literature [**ramos2015towards**, **bazzan2016multiagent**, **bazzan2014evolutionary**, **grunitzki2014individual**]. This is a larger network with 4 origin-destination pairs, 102 cells (and 2 queues) and 41 junctions (28 junctions with only one incoming and one outgoing cell, and 13 more general junctions). We set the total demand to be $\bar{\lambda}^{\mathrm{h}}+\bar{\lambda}^{\mathrm{a}}=3.46$ cars per second, distributed equally to the 4 origin-destination pairs in expectation. As there are 1752 possible different simple paths that vehicles could be taking, our action space is 1752 dimensional. While such an optimization is still possible with powerful computation resources, it might be unnecessary because an optimal solution is unlikely to utilize the paths that traverse too many cells. We therefore restrict our action space to the 10 shortest paths (with respect to the free-flow latencies) between each origin and destination, and so adopt a 40-dimensional action space. We keep the other experiment parameters the same as the small network experiment above.

Due to the network size and the computation cost to simulate the OW network, the MPC-based controller does not produce any useful results in a reasonable time as explained before. We instead implemented the greedy optimization method of **krichene2018social** as a baseline. Specifically we used a genetic algorithm for the optimization with a constraint on the run time of one minute, as it is an online algorithm. It is important to note that RL policy makes a routing decision within a millisecond during test time. We compare the RL controller with this greedy method and the selfish routing.

Fig. 6 shows the number of cars in the network over time (mean $\pm$ standard error over 100 simulations). Again, the selfish routing and the greedy optimization method of [**krichene2018social**] suffer from linearly growing queues, while RL controller is able to stabilize the queues and keeps the network uncongested even though the network may start from a congested state. Furthermore, we check whether the reduced action space is really sufficient. We observe that, over 100 episodes, 98.92% of the autonomous vehicles were routed to the paths that are faster than the fastest path that is not in the action space.

To analyze the performance RL controller in comparison with the optimal equilibrium, we now move to parallel networks.

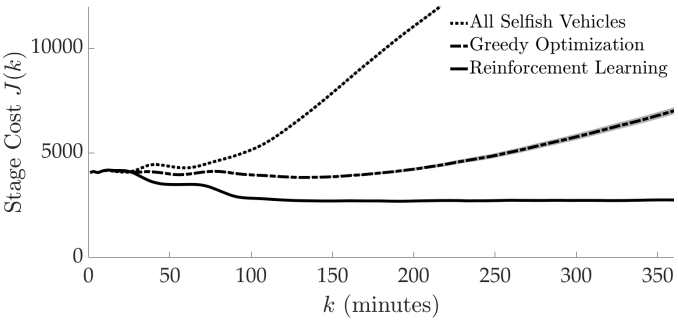

Fig. 6: Time vs. number of cars under selfish, greedy and RL routing on OW network.

B. Parallel Networks

We consider a parallel network from downtown Los Angeles to the San Fernando Valley with 3 paths. The highway numbers and the approximated parameter tuples (length, number of lanes, speed limit) are:

1) 110N (5 miles, 3 lanes, 60 mph); 101N (10 miles, 3 lanes for 5 miles then 2 lanes, 60 mph)
2) 10E (5 miles, 4 lanes, 75 mph); 5N (10 miles, 4 lanes, 75 mph); 134W (5 miles, 3 lanes, 75 mph)
3) 10W; 405N (both 10 miles, 4 lanes, 75 mph); 101S (5 miles, 3 lanes, 75 mph)

As the cells are now not shared between the paths, we employ better latency estimates for human choice dynamics: we compute them as the actual latencies that would occur if there were no accidents and no more demand into the network.

We perform 3 sets of experiments. In the first two, we disable accidents and analyze the effects of varying the number of paths and autonomy. As the shortest path has 15 cells, we exclude MPC-based controller from our analysis as it is computationally prohibitive to adopt a receding horizon longer than 15 minutes.

Varying number of paths. We first vary the number of paths $|\mathcal{P}| \in \{2, 3, 4\}$ by duplicating, or removing, the third path. We set the autonomy level of the demand $\bar{\alpha} = 0.6$, and $\bar{\lambda}^{\mathrm{h}} + \bar{\lambda}^{\mathrm{a}}$ to be 95% of the maximum capacity under this autonomy level. We plot learning curves in Fig. 7 (a). It can be seen that even with $|\mathcal{P}| = 4$ when observation space is 144-dimensional, the agent successfully learns routing within 40 million time steps. With randomized initial states, the agents learn routing policies that achieve nearly as good as optimal equilibrium for all $|\mathcal{P}| \in \{2, 3, 4\}$. In Fig. 7 (b), we plot the number of cars (mean $\pm$ standard error over 100 simulations) in the system over time. While selfish routing causes congestion by creating linearly growing queues when $|\mathcal{P}| > 2$, RL policies successfully stabilize queues and even reach car numbers of optimal equilibria.

Varying autonomy. We take $|\mathcal{P}| = 3$ and vary the autonomy of demand $\bar{\alpha} \in \{0.4, 0.5, 0.6, 0.7\}$ without changing the total demand $\bar{\lambda}^{\mathrm{h}} + \bar{\lambda}^{\mathrm{a}}$. Note the demands are infeasible when $\bar{\alpha} \in \{0.4, 0.5\}$. In Fig. 8 (a), we plot the number of cars (mean $\pm$ standard error over 100 simulations) in the system over time. The result is similar to the previous experiment when the demand is feasible. With infeasible demand, RL agent keeps a queue that is only marginally longer than the queue that optimal equilibrium would create. On the other hand, selfish routing grows the queue with much faster rates. These experiments show RL policy successfully handles random initializations.

Accidents. In the third set, we fix $|\mathcal{P}| = 3$ and $\bar{\alpha} = 0.6$ for the same total average demand and enable accidents. As before, the expected frequency of accidents is 1 per 100 minutes, and clearing out an accident takes 30 minutes on average. Fig. 8 (b) shows the RL policy successfully handles accidents, indicating a good generalization performance by the RL controller. To give a clearer picture, we provide the space-time diagrams and the detailed information about the system states of a sample run in Figs. 9 and 10, respectively. Fig. 9 shows that selfish routing causes congestion by not utilizing the third route, whereas RL can avoid congestion and handle accidents. Fig. 10 shows the number of cars in each cell as well as the queue lengths over time. The small oscillations, which occur even after the effect of the accidents disappear (between third and fourth hours), are due to noisy demand and the discretization of cells. With selfish routing, the vehicles use the longest path only when there is an accident in another path (around first and third hours) or the other two paths are congested (third and fifth hours). In contrast, RL makes good use of the network and leads to altruistic behavior. It also handles the accidents by effectively altering the routing of autonomous cars (around third hour, autonomous cars start using the first route until the accident in the third route is cleared). Hence, it manages to stabilize the queue and prevent congestion. We provide video visualizations of this run at https://youtu.be/XwdSJuUb09o.

VI. Conclusion

Summary. We presented a framework for understanding a dynamic traffic network shared between selfish human drivers and controllable autonomous cars. We show, using deep RL, we can find a policy to minimize the average travel time experienced by users of the network. We develop theoretical results to describe and calculate the best equilibria that can exist and empirically show that our policy reaches the best possible equilibrium performance in parallel networks. Further, we provide case studies showing how the training period scales with the number of paths, and we show our control policy is empirically robust to accidents and stochastic demand.

Limitations. We used the number of cars in each cell as predictive features for RL training. Although this makes the state space dimensionality grow only linearly with the number of cells, it may not be scalable to much larger traffic networks. Moreover, the action space grows linearly with the number of source-destination pairs, also impacting the scalability of the algorithm.

Future work. This work opens up many future directions for research, including using multi-agent reinforcement learning to model autonomous vehicles with competitive goals and/or en route decision making ability, and improving how the training time scales with the complexity of the network. Another interesting future work is to investigate how an RL policy can be deployed and the simulation imperfections (including the dependency on the simulated human choice dynamics) can be alleviated by collecting online data using sensors from the real traffic network.

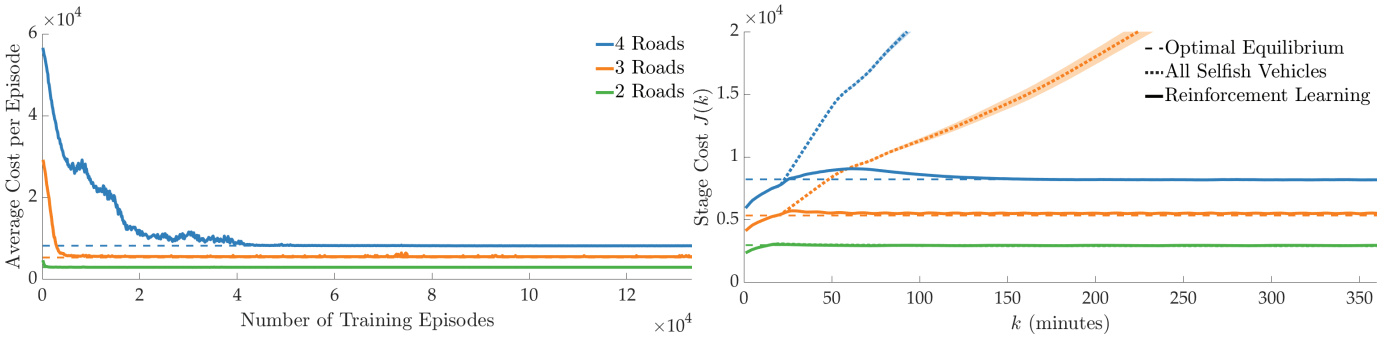

Fig. 7: Varying number of paths. **(a)** Average number of cars in the system per episode during RL training. **(b)** Time vs. number of cars in the system for the comparison of selfish and RL routing in parallel networks.

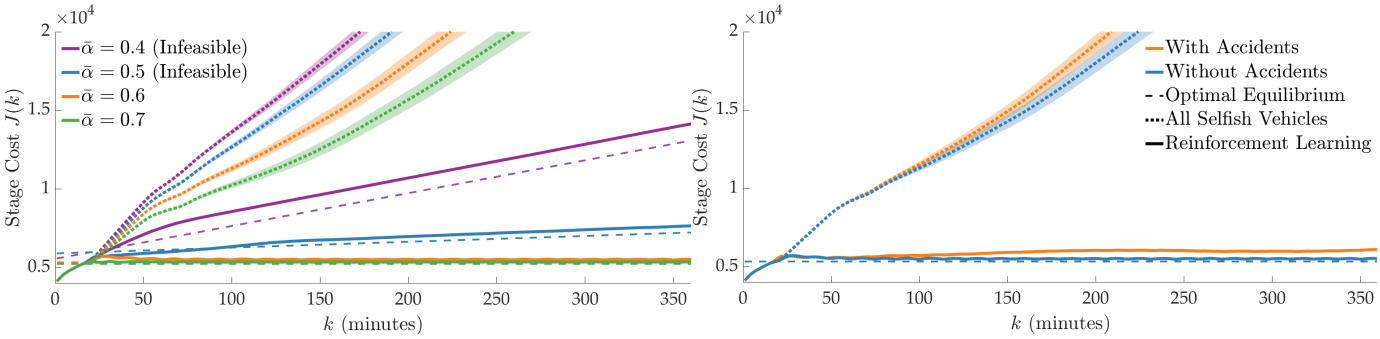

Fig. 8: **(a)** Varying autonomy. **(b)** Varying the presence of accidents and noise in the demand.

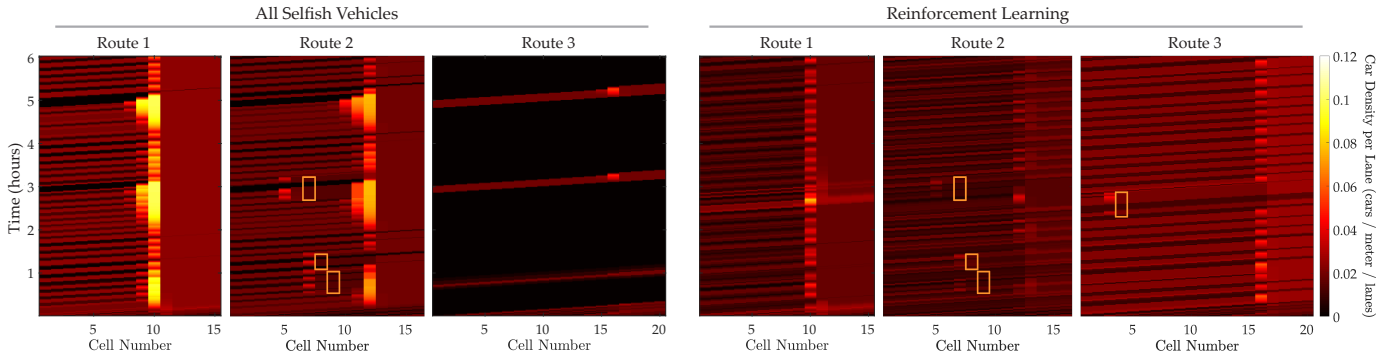

Fig. 9: Space-time diagrams on a parallel traffic network with accidents and noisy demand. Orange rectangles represent accidents.

ACKNOWLEDGMENTS

This work was supported by NSF grant #1953032 and Toyota. Toyota Research Institute (TRI) provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

VII. APPENDIX

A. *Summary of notation*

See Table I.

B. *Proofs for Section IV-C*

Proof of Lemma 1. By definition, at equilibrium, the number of vehicles in each cell i in $\mathcal{I}_p$, $n_i^{\mathrm{a}}(k)$ and $n_i^{\mathrm{a}}(k)$ is constant for all times k. Since by definition the incoming flow is also constant, by the definition of the sending and receiving functions, constant cell densities implies constant flows. By (3), a constant density also implies that the incoming and outgoing flow in each cell are equal. This means that all cells will have the same incoming flow as the first cell. Further, we know that since the density of autonomous vehicles is constant over time, incoming and outgoing autonomy levels are equal. Accordingly, if cell i' is the cell immediately upstream of i, then

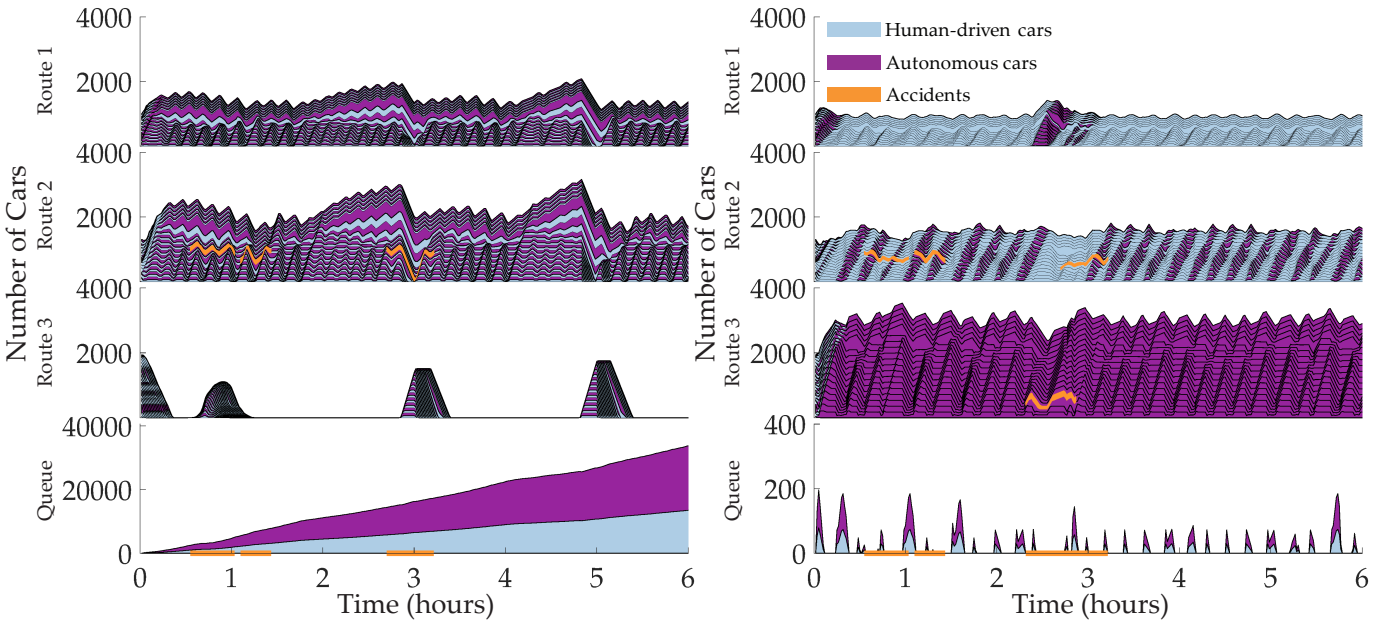

Fig. 10: The network under perturbations due to accidents and noisy demand. For each path and time step, from bottom to top, the stacked color segments show the number of cars in the cells from origin to the destination. Congestion occurs only upstream to the bottlenecks. **(a)** Selfish routing. **(b)** RL routing.

TABLE I: Summary of Notation

p	Path index	unitless
$\mathcal{P}$	Set of paths in the network	set of paths
i	Cell index	unitless
$\mathcal{I}$	set of cells in the network	set of cells
$\mathcal{I}_p$	set of cells in path p	set of cells
$\mathcal{U}_i$	set of cells upstream of cell i	set of cells
$\bar{v}_i$	Free-flow velocity of cell i	cells/time step
b_i	Number of lanes of cell i	unitless
h_i^{h} (h_i^{a})	Nominal vehicle headway on cell i	cells/vehicle
n_i^{h} (n_i^{a})	Density of vehicles on cell i	vehicles/cell
n_i	Total vehicle density on cell i	vehicles/cell
f_i^{h} (f_i^{a})	Flow of vehicles from cell i	vehicles/time step
y_i^{h} (y_i^{a})	Hum. (aut.) veh flow into cell i	vehicles/time step
α_i	Autonomy level of cell i	unitless
$\tilde{n}_i(\alpha)$	Critical density of cell i, at aut. α	vehicles/cell
$\bar{n}_i$	Jam (maximum) density of cell i	vehicles/cell
$\bar{F}_i(\alpha)$	Capacity of cell i, at aut. α	vehicles/time step
$w_i(\alpha)$	Shockwave speed of cell i, at aut. α	cells/time step
k	Time index	unitless
$\ell_p(k)$	Latency of path p if starting at time k	time steps
$q_i(k)$	Priority for cell i at a merge at time k	unitless
$\mu_i^{\mathrm{h}}(p,k)$ (μ^{a})	Frac. of hum. (aut.) vehs in i on p at k	unitless
$\beta_i^{\mathrm{h}}(i',k)$ (β^{a})	Frac. of hum. (aut.) vehs $i \to i'$ at k	unitless
$J(k)$	Stage cost at time k	vehicles
m_p^{b} (m_p^{n})	# of (non)bottleneck cells on path p	cells
b^{b} (b^{n})	# of lanes in (non)bottleneck cells on p	unitless
r_p	$:= b^{\mathrm{b}}/b^{\mathrm{n}}$	unitless
γ_p	Number of congested cells on path p	cells

$\alpha_{i'}(k)f_{i'}(k) = \alpha_i(k)f_i(k)$. Since we also have $f_{i'}(k) = f_i(k)$, this implies that $\alpha_{i'}(k) = \alpha_i(k)$. Therefore the autonomy level of all cells is the same. Let us denote this uniform autonomy level α_p. Let the index of the first cell in the path be 0. Then, $\bar{\lambda}_p^{\mathrm{h}} + \bar{\lambda}_p^{\mathrm{a}} = f_0$ and $\bar{\lambda}_p^{\mathrm{a}} = \alpha_p f_0$. Combining these two expressions, we find $\alpha_p = \bar{\lambda}_p^{\mathrm{a}}/(\bar{\lambda}_p^{\mathrm{h}} + \bar{\lambda}_p^{\mathrm{a}})$. □

Proof of Lemma 4. Under Assumption 2, no two paths have the same free-flow latency. With Proposition 1, this implies that if an equilibrium has a used path with no congestion, it must be the used path with greatest free-flow latency, as otherwise all used paths would not have the same latency. Therefore, if an equilibrium routing with positive flow on paths $[p]$ has a path in free-flow, it must be path p. Otherwise, we can construct an equilibrium with the same demand that has path p in free-flow. Recall that the latency on paths in equilibrium is increasing with the length of the congested portion of the path, $\gamma_{p'}$, and $\gamma_{p'} = 0$ corresponds to an uncongested path. If all paths are congested, we consider decreasing the length of congestion on all paths simulatenously, at rates which keep the path latencies equal. This continues until path p becomes completely uncongested. This construction proves the lemma. □

C. Overview of Proximal Policy Optimization (PPO)

In this section, we give a brief overview of the PPO method [**schulman2017proximal**] we used for training our deep reinforcement learning model. We first start with formalizing the problem. We then introduce the policy gradients and the details of PPO. To keep the notation consistent with the reinforcement learning literature, we abuse the notation for some variables. Hence, this section of the appendix is written in a standalone way, and the variables should not be confused with the notation introduced in the main paper (e.g. f is going to denote the transition distribution of the system as introduced below, instead of flow values as in the main paper).

Problem Setting. We consider a sequential decision making problem in a Markov decision process (MDP) represented by a tuple $(\mathcal{S}, \mathcal{A}, f, T, r, \gamma)$, where $\mathcal{S}$ is the set of states. $\mathcal{A}$ denotes the set of actions, and the system transitions with respect to the transition distribution $f : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. For example, if $f(s, a, s') = p$, this means taking action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$ transitions the system into state s' with probability p. Next, T denotes the horizon of the system, i.e., the process

gets completed after T time steps. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ maps state-actions to reward values. The decision maker is then trying to maximize the cumulative reward over T time steps by only observing the observations (not states). Finally γ is a discount factor that sets how much priority we give to optimizing earlier rewards in the system.

Let us now describe how we formulate a transportation network with the CTM model as an MDP in this paper. The state of the network is fully defined by the following information:

- Location of each vehicle (which cell or queue it is in),
- Type of each vehicle (human-driven or autonomous),
- Accident information (where and when it happened), and
- Planned path of each vehicle (which cells it is going to traverse).

In our model, we assumed the first three items in the above list are available as observations. While this breaks the Markov assumption, deep RL techniques often perform well in partially observable MDPs, too. So our deep RL policy is trying to make its decisions based only on those first three observations, and the non-observability of the planned paths increases the stochasticity of the problem. The action set of the decision maker is defined by the set of available routing paths of autonomous vehicles. The transition distribution follows the dynamics of CTM, human choice dynamics, as well as the accidents which also introduce stochasticity into the system. Finally, as a reward function, one can think of using the negative of the number of cars in the system as a proxy to negative of overall latency in the network.

Policy Gradients. To solve this problem using deep neural networks, we model the decision-maker agent with a stochastic policy π_θ parameterized with θ (e.g. weights of the neural network), such that $\pi_\theta(a \mid s)$ gives the probability of taking action a when observing state s. The goal of the agent is to maximize the expected cumulative discounted reward:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right]$$

where τ denotes a trajectory $(s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T)$ in the system. The discount factor is to improve robustness and to reduce susceptibility against high variance. We can equivalently write this objective as:

$$J(\theta) = \int_\Xi \pi_\theta(\tau) r(\tau) d\tau$$

where Ξ is the set of all possible trajectories, $\pi_\theta(\tau)$ is the probability of trajectory τ under policy π_θ, and $r(\tau)$ is the cumulative discounted reward of trajectory τ. The idea in policy gradients is to take gradient steps to maximize this quantity by optimizing θ:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int_\Xi \pi_\theta(\tau) r(\tau) d\tau \\
&= \int_\Xi \nabla_\theta \pi_\theta(\tau) \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} r(\tau) d\tau \\
&= \int_\Xi \pi_\theta(\tau) r(\tau) \nabla_\theta \log \pi_\theta(\tau) d\tau \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[r(\tau) \nabla_\theta \log \pi_\theta(\tau)\right]
\end{aligned}$$

which we can efficiently approximate by sampling trajectories using the policy.

Unfortunately, this vanilla policy gradient method is not robust against variance (due to stochasticity in the environment and trajectory sampling) and suffers from data-inefficiency. In recent years, several works have developed alternative ways to approximate the gradients. One such idea is based on using baselines to reduce variance:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^\tau \mid s_t^\tau) \hat{A}_t^\tau\right]$$

where $\hat{A}$ is called the estimated advantage function, which is usually defined as $G_t^\tau - V(s_t^\tau)$, where G_t^τ is the cumulative discounted reward of the trajectory τ after (and including) time step t, and $V(s_t^\tau)$ is some baseline that quantifies the value of state s_t^τ. This new equation for $\nabla_\theta J(\theta)$ holds due to the Markov assumption and that the baseline is independent from the policy parameter θ.

Having presented the policy gradients and the use of baselines for variance reduction, we are now ready to give an overview of PPO.

Proximal Policy Optimization (PPO). PPO further improves the robustness and data-efficiency of policy gradient methods by using a surrogate objective that prevents the policy from being updated with large deviations. Instead of the usual objective $\mathbb{E}_{\tau\sim\pi_\theta}\left[\log \pi_\theta(a_t^\tau \mid s_t^\tau)\hat{A}_t^\tau\right]$, PPO uses the following objective:

$$J_1(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\min(g_t^\tau(\theta)\hat{A}_t^\tau, \text{clip}(g_t^\tau(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^\tau)\right]$$

where

$$g_t^\tau(\theta) = \frac{\pi_\theta(a_t^\tau \mid s_t^\tau)}{\pi_{\theta_{\text{old}}}(a_t^\tau \mid s_t^\tau)} \text{ and } \text{clip}(x, \epsilon_1, \epsilon_2) = \begin{cases} \epsilon_1 & x < \epsilon_1, \\ x & \epsilon_1 \le x \le \epsilon_2, \\ \epsilon_2 & \text{otherwise.} \end{cases}$$

In addition to $J_1(\theta)$, PPO uses two more objective functions and converts the problem into a multi-objective optimization problem. The first additional objective is for the baseline $V(s_t^\tau)$. Specifically, PPO learns a parameterized value function V_ϕ in a supervised way to minimize $(V_\phi(s_t^\tau) - V_t^{\text{target}})^2$ where V_t^{target} is calculated using the sampled trajectories as a sum of discounted rewards after (and including) time step t. It should be noted that this does not make $G_t^\tau - V_\phi(s_t^\tau) = 0$, because $V_\phi(s_t^\tau)$ is an estimate of the true value function and is updated after the computation of the estimated advantage. Therefore,

$$J_2(\phi) = -\mathbb{E}_{\tau\sim\pi_\theta}\left[V_\phi(s_t^\tau) - V_t^{\text{target}}\right] .$$

Finally, PPO uses an entropy bonus (inspired by [**mnih2016asynchronous**]) to ensure sufficient exploration:

$$J_3(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} H(\pi_\theta(\cdot \mid s_t^\tau)) ,$$

where H is information entropy. At the end, PPO tries to solve:

$$\text{maximize}_{\theta,\phi} \quad J_1(\theta) + J_2(\phi) + cJ_3(\theta)$$

where c is the coefficient for the entropy term.

D. Experiment details

In implementation, we used $J(k) - J(k-1)$ as a proxy cost for time step k, where $J(0) = 0$.

Below are the set of hyperparameters we used for PPO. We refer to Section VII-C and [**schulman2017proximal**] for the definitions of PPO-specific parameters. While this set yields good results as we presented in the paper, a careful tuning may improve the performance.

- Number of Time Steps: 40 million
- Number of Actors: 32 (32 CPUs in parallel)
- Time Steps per Episode During Training: 300
- Time Steps per Actor Batch: 1200
- ϵ for Clipping in the Surrogate Objective: 0.2
- Optimization Step Size (OSS): 3×10^{-4}
- Annealing for ϵ (Clipping) and OSS: Linear (down to 0)
- Entropy Coefficient: 0.005
- Number of Optimization Epochs: 5
- Optimization Batch Size: 64
- γ for Advantage Estimation: 0.99
- λ for Advantage Estimation: 0.95
- ϵ for Adam Optimization: 10^{-5}

Finally, we report the training times (for 40 million time steps) and the number of time steps of empirical convergence (in terms of reward value) for each RL policy in Table II. In test time, RL policies produce an action in under 1 ms.

TABLE II: Training and Convergence Times

Policy	**Training Time**	**Time Step of Convergence**
Simple General Network	10.0 hours	26.3 million
OW Network	253.1 hours	31.0 million
$\lvert\mathcal{P}\rvert = 2$	22.2 hours	0.7 million
$\lvert\mathcal{P}\rvert = 3$	38.9 hours	10.0 million
$\lvert\mathcal{P}\rvert = 3$, w/ accidents	40.5 hours	22.0 million
$\lvert\mathcal{P}\rvert = 3$, $\bar{\alpha} = 0.4$	50.6 hours	25.5 million
$\lvert\mathcal{P}\rvert = 3$, $\bar{\alpha} = 0.5$	43.1 hours	19.3 million
$\lvert\mathcal{P}\rvert = 3$, $\bar{\alpha} = 0.7$	38.6 hours	6.6 million
$\lvert\mathcal{P}\rvert = 4$	101.4 hours	23.3 million