Thesis

Rethinking the Design of Human-Centric AI Systems for Deployment in Transportation

Rex Chen

CMU-S3D-25-107

July 2024

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Fei Fang	(Co-chair)	Carnegie Mellon University
Norman Sadeh	(Co-chair)	Carnegie Mellon University
Sean Qian		Carnegie Mellon University
Matteo Pozzi		Carnegie Mellon University
Ryan Shi		University of Pittsburgh
Peter Stone		The University of Texas at Austin

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Societal Computing.

This research was supported by a NSERC Postgraduate Scholarship (Doctoral), a Mobility21 grant, the Tang Family Endowed Innovation Fund, and a SCS Presidential Graduate Fellowship.

Abstract

Artificial intelligence (AI) has had a transformative impact in improving the efficiency, safety, and accessibility of transportation systems. Successes in existing deployments have, in turn, driven substantial interest from the AI literature in transportation as an application domain. Deep learning and multi-agent learning algorithms have shown particularly strong performance on transportation problems in simulation-based evaluations. However, many of the state-of-the-art algorithms from the AI literature have never achieved physical deployment.

In this thesis, I argue that a barrier to deployment for many advanced AI technologies in transportation lies in the fact that they are often divorced from the perceptions, preferences, pain points, and priorities of human stakeholders — and thus cannot be reliably integrated into existing transportation systems. Using gig driving and traffic signal control as two representative problems, I explore how AI algorithms used for modelling and decision-making in transportation systems can better address challenges that arise from interactions with people and systems in deployment.

More specifically, I consider the four key challenges of:

(1) *uncertainty* in present and projected traffic conditions;
(2) *heterogeneity* among users and deployment contexts;
(3) *assurance* in terms of the understandability and safety of algorithms; and
(4) *coordination* between individuals and systems.

Through my contributions in this thesis, I show that designing AI technologies to better address these challenges leads to technically novel algorithms, and I suggest that the path to deployment for AI technologies in transportation lies in building them with people in mind at every stage of the data-to-deployment pipeline.

Acknowledgments

TBD

Contents

Chapter 1

Introduction

Emerging technologies based on *artificial intelligence* (AI) have revolutionised the state of modern transportation systems. Over the past decade, AI technologies have been deployed at exponentially increasing scales to optimise traffic signals for reducing congestion and emissions [197, 343]; to dispatch gig drivers for enhancing the mobility of people and goods [138, 206]; to streamline public transportation and freight logistics for improving utilisation and efficiency [145, 271]; and to control connected and autonomous vehicles for making individual transport safer and more accessible [114, 213]. In these applications, AI has mitigated systematic inefficiencies caused by suboptimal practices and insufficient capacity in existing transportation infrastructure [288, 322].

Key to these successes has been the proliferation of *deep learning*, which has enabled AI algorithms to achieve performance beyond that of humans or even of non-deep learning algorithms [314]. Concomitant advances in computing infrastructure have also allowed AI technologies to adapt to rapid change by ingesting data at massive scales [281]. These results have fuelled the growth of transportation as an application of AI, alongside optimistic attitudes among forward-thinking stakeholders who believe in AI's capabilities to improve the status quo [284].

Accordingly, transportation has also attracted significant attention from the AI literature as an application domain; researchers have applied AI algorithms to solve various problems based on transportation systems [335]. These problems have ranged from prediction, forecasting, and pattern recognition to control, planning, and optimisation [153], based on applications in ground (including basic infrastructure, vehicle control, fleet management, and public transit) [102, 221, 279], air [81, 299], and maritime [266] transportation. For many of these problems, AI algorithms have demonstrated strong capabilities in simulation-based experimental evaluations (e.g. [14, 223]).

However, a disconnect between theory and practice exists. Many of state-of-the-art algorithms developed by the AI literature have never achieved deployment in practice, despite their seemingly strong performance. Even when they have been deployed, their potentials have largely remained unrealised. In particular, there are many stakeholders who remain concerned about the possible impacts of these AU algorithms on the efficiency, equity, and ethics of transportation systems [284]. In this thesis, I ask: Why does the wheel of progress turn so slowly? What challenges are preventing stakeholders from being willing to deploy AI technologies for transportation? What technical advances in AI are needed to pave the way towards successful deployments, so that the promising performance in experimental evaluations can be translated into tangible benefits?

1.1 Problem Domains

Within the broad space of AI for transportation, I address these motivating questions by narrowly focusing on two specific applications of AI:

- *Gig driving* (reviewed in Section 2.1), where platforms dispatch a pool of drivers to provide on-demand transportation of passengers or orders
- *Traffic signal control* (TSC; reviewed in Section 2.2), where controllers set the light sequences of a series of traffic signals to enable vehicles to pass through intersections

In Table 1.1, I summarise the characteristics of gig driving and traffic signal control. While every problem in transportation entails an idiosyncratic set of objectives and challenges, some of these characteristics are shared with other problems, and my goal is accordingly to generalise insights that can be derived from these two representative problems.

Gig driving	Traffic signal control
Similarities	
Multi-agent: Usually involves coordinating many drivers / intersections **Sequential decision-making**: Need to keep taking gigs / signalling phases	
Differences	
Individual-level control: Each driver makes their own self-interested decisions	**System-level control**: Control could be centralised but is distributed for efficiency
Mixed competitive-cooperative: Various different drivers and platforms may collaborate or conflict with each other	**Cooperative**: All intersection agents seek to minimise the same performance metrics across the entire road network
Opacity: Platforms are revenue maximisers and hide information from their users	**Transparency**: Stakeholders need traceability in order to pursue social benefits

Table 1.1: Comparison of problem characteristics for gig driving and traffic signal control

Both of these problems have been the subject of vigorous study in the AI literature, such that many AI-based solutions have been proposed, but few of them have been deployed. For gig driving, dispatching mechanisms with various theoretical guarantees have been designed for platforms, but their practical implementation more often relies on heuristics [276] that lead to suboptimal outcomes for drivers [246]. For TSC, reinforcement learning (RL) algorithms easily control and coordinate between dozens, hundreds, and even thousands of traffic signals in simulations [54]. Yet, after a decade of advances in RL, the only deployment of these algorithms that I am aware of is limited to three intersections considered in isolation from each other [71].

1.2 Thesis Statement

My goal in this thesis is to *integrate the perceptions, preferences, pain points, and priorities of stakeholders into the design of AI technologies for transportation*. In doing so, I address what I consider to be the most critical challenge that hinders the practical applicability of AI technologies in transportation: not their performance, but rather their capacity to be embedded into complex, human-centric deployment contexts. I hypothesise that this is where the AI literature has fallen short. By optimising for gains in performance rather than deployability, many state-of-the-art AI algorithms have become divorced from the practical deployment challenges faced by stakeholders. In this thesis, I address research questions relating to the following four deployment challenges:

- **Uncertainty**. How can AI technologies in transportation achieve robust levels of performance when spatiotemporal variability in the movement of people makes it difficult to observe or predict traffic conditions? This challenge arises in the transition from simulation to deployment: through varying traffic patterns and sensor imperfections in traffic signal control, and through the on-demand nature of gigs and information asymmetry in gig driving. If not accounted for, such sources of uncertainty could degrade the performance of AI technologies from simulation-based evaluations when they are deployed.
- **Heterogeneity**. How can AI technologies in transportation be broadly applied in the presence of variation between end-users and deployment contexts, and how might this be hindered by assumptions made during their design? This challenge arises during the generalisation of AI technologies: with road users or gig drivers having different personalities, motivations, and vehicle types that lead them to react to AI-driven decisions differently, and across road networks and cities with different characteristics. Abstracting out such variation can significantly impact the outcomes of AI technologies when they are deployed.
- **Assurance**. How can AI technologies in transportation present their decisions to end-users and stakeholders in an understandable, traceable fashion? This challenge arises during design and after deployment: decisions made by deployed AI technologies (e.g. signal plans in traffic signal control or work schedules in gig driving) need to be presented in an understandable and safe manner. On one hand, they should align with stakeholders' mental models acquired through prior experience. Meanwhile, they should also provide guarantees in terms of safety and opportunity costs, lest they become difficult for stakeholders to trust.
- **Coordination**. How can AI technologies in transportation reason about interactions between individual agents (both human and AI) to achieve equitable system-level outcomes? This challenge arises inherently in the design process: traffic signal control requires coordinated timings for intersections, and gig driving requires coordination between individual drivers' movements. Contrary to assumptions in prior work, it is not evident that such coordinated behaviour can be emergently learned by AI technologies, considering the diversity of objectives among individuals at different levels of hierarchical control in the real world.

However, theoretical and practical contributions in gig driving and TSC are not mutually exclusive. This thesis explores many dimensions by which stakeholder-centric considerations can be incorporated into the design and construction of AI technologies to address these deployment challenges. Such dimensions include fostering their perceptions of trust and reliance on AI (Chapter 3),

learning from their practices and pain points in established deployments (Chapter 2), accounting for their distinct patterns of behaviour (Chapter 4), using their prior knowledge to fill in unknowns (Chapter 5), enforcing practical constraints that they can depend on for safety (Chapter 6), and creating predictable model structures that they can interpret (Chapter 7). My solutions to these deployment challenges lead to technical novelties that also generalise to other applications of AI technologies in — and beyond — transportation.

I argue that these practical considerations about people must not be an afterthought, but instead must be integrated end-to-end throughout the design, implementation, and evaluation process. By their involvement, stakeholders in transportation can be assured that their pain points will be addressed — that the purpose of AI technologies is not to substitute or frustrate them, but to complement and enhance their work [211, 300]. It is only then that AI technologies in transportation will move from the cutting edge of the technical literature to physical deployments at scale, thus achieving their potential to better the fast-changing transportation systems of today's world.

1.3 Contributions

In this thesis, my contributions include:

- **Chapter 2**: A literature review of these four deployment challenges
- **Chapter 3**: A schedule recommendation tool for gig drivers, and a longitudinal user study showing that different framings of *uncertainty* impact trust in and reliance on the tool
- **Chapter 4**: An empirical evaluation of how adding *heterogeneity* in driver behavioural models can significantly affect traffic simulators used to train RL algorithms
- **Chapter 5**: An end-to-end algorithmic pipeline for constructing a high-fidelity traffic simulation, starting from *uncertain* and *heterogeneous* detector data
- **Chapter 6**: A multi-agent RL algorithm for TSC that retains state-of-the-art *coordinated* performance while providing *assurances* by imposing signal plan constraints
- **Chapter 7**: A multi-agent imitation learning algorithm that efficiently distils RL-based TSC policies into *coordinated* decision trees that provide *assurances* through interpretability

With Chapters 4 to 7, I study these deployment challenges throughout different stages of the pipeline of deploying RL policies for TSC: choosing a simulator, building a simulation, training the RL policy, and lastly imitating the RL policy using an interpretable surrogate.

Chapter 2

The Real Deal

A Review of Gig Driving, Traffic Signal Control, Reinforcement Learning, and Deploying AI for Transportation

Domain: Gig driving, traffic signal control
Challenges: Uncertainty, heterogeneity, interpretability, coordination

In Chapter 1, I introduced *gig driving* and *traffic signal control* (TSC) as two transportation problems that I focus on as deployment contexts of AI technologies. This chapter provides background on both problems. Section 2.1 and Section 2.2 begin with overviews of gig driving and TSC from the perspective of stakeholders — gig platforms, gig drivers, and traffic engineers — rather than from the perspective of the AI literature. I show that pain points do exist in the state of practice: information asymmetry leads to unpredictability and inequity in gig driving, while imperfect detection and control reduce throughput in TSC. Next, in Section 2.3, I introduce *reinforcement learning* (RL), the class of AI algorithms for TSC that I consider in Chapters 6 and 7. Lastly, in Section 2.4, I describe how the deployment challenges of uncertainty, heterogeneity, interpretability, and coordination from Chapter 1 occur in gig driving and TSC. I also discuss how existing methods from the AI literature have addressed these challenges for both problems, how they have been inadequate, and what work still remains to be done.

This chapter was adapted from text in various publications, but a significant portion was published as a review paper that surveyed deployment challenges of RL for TSC, at the International Workshop on Agents in Traffic and Transportation (ATT) at IJCAI in 2022 [58].

2.1 Gig Driving

2.1.1 Problem Formulation

Gig drivers are independent contractors who use their personal vehicles to provide on-demand transportation of either passengers (*ridesourcing*) and restaurant or grocery orders (*food delivery*)

from app platforms such as Uber, Lyft, DoorDash, and Instacart. I consider ridesourcing and food delivery as a unified problem, not only due to similarities in their problem formulations [147] but also because drivers view them interchangeably [196] (e.g. Uber and its food delivery business, Uber Eats, are mutually cannibalistic platforms [65]).

In gig driving, there are interactions between three distinct groups of stakeholders: the customers who request rides or deliveries, the drivers who service these requests, and the platforms which mediate their interactions (Figure 2.1). Customers initiate by sending their requests and payments directly to the platforms. The platforms then attempt to match the set of requests with the set of available drivers. Each driver is then shown a list of requests that the platform recommends, and is free to accept or decline each one. If they accept, they fulfill the request; if they decline, the platform falls back to an alternative assignment. Lastly, platforms distribute payments to drivers. From the perspective of the platforms, this is an iterative weighted matching problem [147].

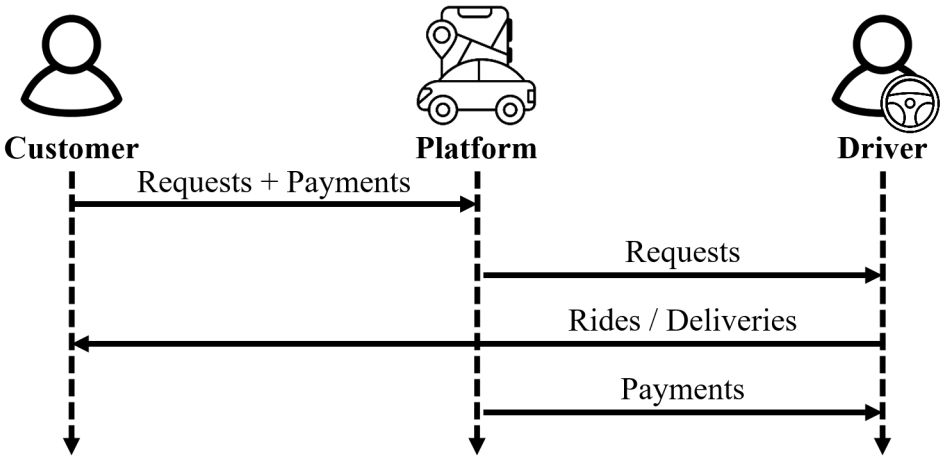


Figure 2.1: Flow of transactions between customers, platforms, and drivers in gig driving.

Although the realisation of this model is a relatively recent phenomenon, gig driving has experienced explosive growth [135] due to the benefits that it provides. For customers, gig driving provides a convenient alternative to both private vehicles and public transport [63, 151, 317, 329]. For drivers, gig driving offers them the flexibility and autonomy to choose when and where to work [56, 196]. Yet, gig driving has also introduced new challenges. Negative interactions between customers and drivers degrade the experiences of both parties [139, 304, 350]. For drivers, the promise of autonomy is also hindered by the opacity of platforms' assignment, pricing, and evaluation mechanisms [125, 196, 340, 366, 410], leading to volatility and systemic inequities in driver outcomes [80, 246]. Lastly, gig driving has created societal externalities. At least in the short term, gig driving will continue to contribute to increased congestion: despite the efforts of platforms to manage their supply and demand, they have not just shifted demand away from other transport modes but also created new demand altogether [138, 317].

2.1.2 AI Algorithms for Gig Platforms

In practice, gig platforms extend the basic problem formulation to both account for and shape fluctuations in supply and demand. One of the most prevalent strategies adopted by these platforms is *dynamic pricing*, referred to as "surge pricing" by Uber, "Prime Time" by Lyft, "Peak Pay" by DoorDash, and "Blitz Pricing" by Postmates [196, 198]. By setting higher prices during peak hours, platforms balance demand by disincentivising customers from making requests, while also incentivising drivers to reposition to high-demand regions and accept requests in exchange for bonuses [220, 276]. Precise implementation details for these mechanisms are generally scant. [276] describes Lyft's "escrow" mechanism, in which convex optimisation problems are solved to prospectively allocate customers' expected Prime Time payments equitably among drivers. However, deployed algorithms likely incorporate various heuristics based on business requirements.

Just as external researchers have little visibility into the inner workings of dynamic pricing algorithms, drivers themselves must contend with both information asymmetries and gamification mechanisms that some perceive to be unfair [437]. Within their limited control, drivers have responded to the algorithmic management exerted by platforms in various ways: switching between platforms [196], engaging in discussion forums [229, 437], and even colluding to induce artificial surges [374]. Nevertheless, the incentives created by these opaque pricing mechanisms still create tangible effects on drivers, including inequitable increases in the level of competition and effort required by drivers [246].

2.1.3 AI Algorithms for Gig Drivers

One line of work in AI for gig driving has focused on gig platforms by designing mechanisms for driver dispatching and payment. These mechanisms aim to achieve various desirable properties such as welfare maximisation and incentive compatibility (i.e. guaranteeing that drivers will accept dispatched trips) [36, 113, 223, 301]. In particular, incentive compatibility is enforced by ensuring that prices offset drivers' opportunity costs [113, 223]. However, these mechanisms generally make unrealistic simplifying assumptions. For instance, drivers in reality rarely have perfect information about even dispatched trips (e.g. drop-off locations are often obscured) [437], much less the opportunity costs of declining them. Even works grounded in particular platforms (such as Ong et al. [276]) fail to consider the complexities that arise from interactions between platforms: how do drivers' opportunity costs change when they can switch to a different platform?

Simultaneously, a complementary line of work has focused on understanding and responding to the needs of drivers; this line of work is closer to the focus that I adopt. Zhang et al. [437] conducted focus groups with ridesourcing drivers to envision possible improvements to gig platforms; one of their core findings was a need for native in-platform data-driven insights that would eliminate the need for third-party tools (e.g. mileage tracking apps). Based on this, Zhang et al. [436] designed "data probes" to help drivers understand how their work patterns and positionalities interact with platforms' management practices. Khan et al. [169] created a measurement suite that quantitatively analyses the dynamic pricing strategies of platforms. Yet, given this abundance of information, there is no work on how to help drivers use this information to make better decisions.

2.2 Traffic Signal Control

2.2.1 Problem Formulation

Traffic signal control (TSC) aims to allocate green time at one or more intersections in a road network to traffic moving in different directions. In this thesis, I consider each intersection to be controlled by an individual, decision-making agent. At each intersection, every *approach* (roadway, e.g. northbound or southbound) is split into *lanes*. Vehicles traversing the intersection can follow different *movements*, each of which is defined by one incoming and one outgoing lane (e.g. northbound left turn or straight through) [122, 399, 455]. A *phase* $\phi \in \Phi = \{\phi_1, \ldots, \phi_P\}$ is a combination of movements that is signalled as a single unit [182]. For efficiency, pairs of non-conflicting movements are often signalled simultaneously (e.g. westbound/eastbound left turn, northbound left turn/straight through) [182, 274, 398].

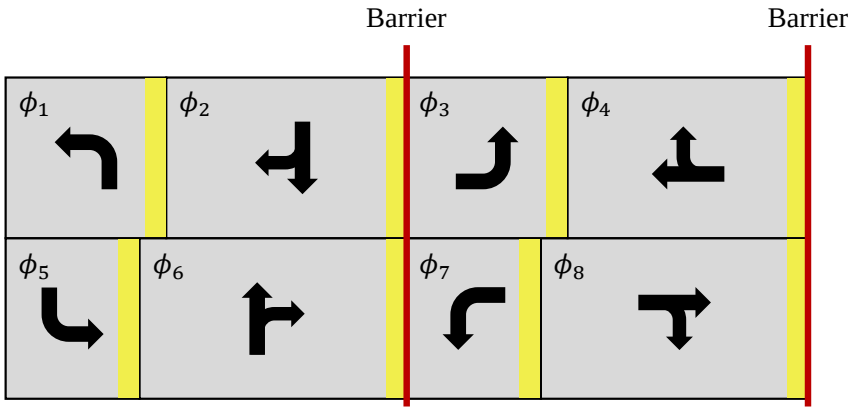


Figure 2.2: An eight-phase NEMA ring barrier diagram for a four-legged intersection.

A *signal plan* for an intersection is defined by a sequence $((\phi_1, t_1)...(\phi_P, t_P))$ of phases and the time allocated to each phase. Under a signal plan, the allocated time for each phase is divided into three intervals: *green* G_i (i.e. a green light, where vehicles can pass freely), *yellow change* Y_i (i.e. a yellow light, where vehicles are warned of a coming change in right of way), and *red clearance* R_i (i.e. a red light, which occurs before the start of the conflicting green interval to ensure that the intersection clears properly) [182]. A typical four-legged intersection follows the eight-phase *NEMA phasing* sequence, which is shown in the *ring barrier diagram* in Figure 2.2. The phase sequence consists of two rings of four phases, each of which serves one road (i.e. $\phi_{1,2,5,6}$ serve the north-south road, while $\phi_{3,4,7,8}$ serve the east-west road). Pairs of non-conflicting phases are usually signalled simultaneously [274, 398]. For instance, ϕ_1 does not create traffic conflicts with either ϕ_5 or ϕ_6, and thus it can overlap with either one; I denote these combinations as $\phi_{1,5}$ and $\phi_{1,6}$. However, the two rings conflict with each other, so the ends of $\phi_{2,6}$ and $\phi_{4,8}$ must be synced.

The task of TSC is to find a signal plan for each intersection that optimises certain *signal performance measures* (SPMs). As vehicles move through the entire road network and improving throughput requires coordinating all of the signals they encounter, TSC is a centralised problem. Some typical SPMs used by stakeholders in TSC include (see also Koohy et al. [181]):

- **Queue length** q (vehs), the number of vehicles that are stopped at an intersection on red.
- **Queue density** q (vehs / m), the number of stopped vehicles normalised by the road length.
- **Pressure** p (vehs), the difference in queue length (or density) between an intersection's incoming and outgoing lanes. Maximising pressure provably maximises throughput [396].
- **Delay** d (veh-hrs), the amount of time that vehicles cumulatively spend stopped at an intersection. This is equal to the difference in cumulative travel time between the theoretical optimum (all vehicles moving at free-flow speed) and the actual time.
- **Flow rate** or **throughput** s (vehs / hr), the rate at which vehicles traverse an intersection.
- **Percentage on green** g, the proportion of traffic that arrives at an intersection during a green light and thus does not incur signalling-induced delay.

2.2.2 Detection Hardware

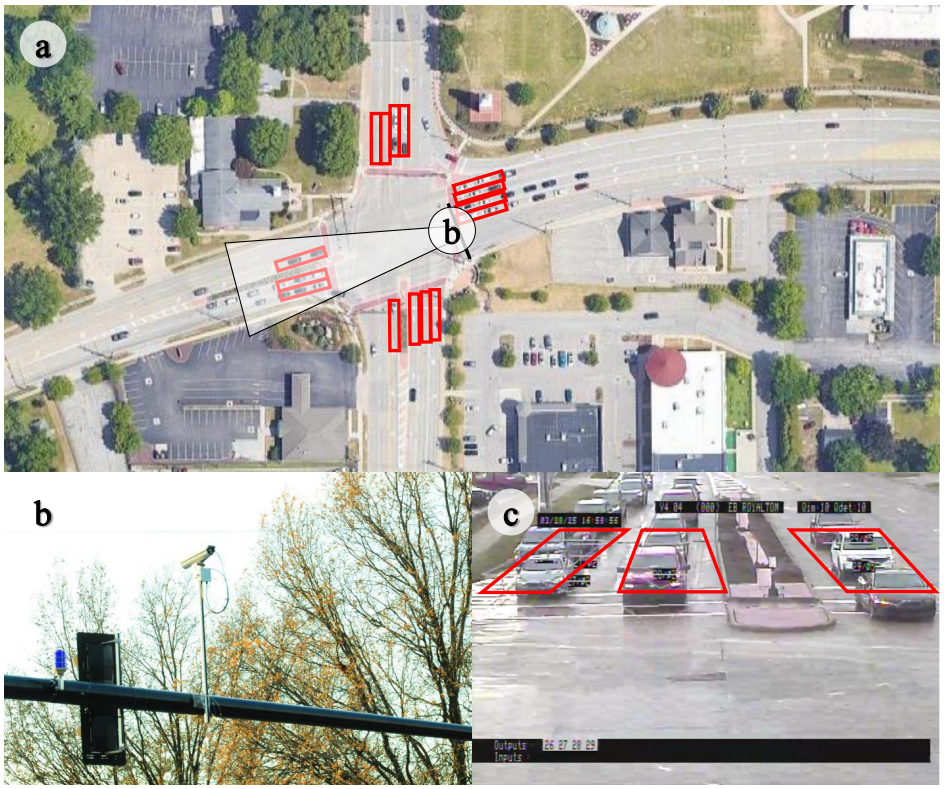


Figure 2.3: Traffic conditions in Strongsville, Ohio, at the intersection of US 42 (north-south) and SR 82 (east-west). (a) Satellite image with shapes of camera detection zones marked in red; (b) photograph of eastbound camera detector; (c) footage from detector with detection zones in red.

How do TSC algorithms interface with the environment? They must first perceive the road state through *detection* hardware, and then alter the signal state accordingly through *control* hardware. Detection hardware usually consists of three types of detectors in the United States.

Loop Detectors An induction loop detector consists of a loop of wire embedded in the pavement (i.e. it is an *intrusive* detector), which is actuated when a vehicle passes over it. Loop detectors are relatively robust to the environment. However, actuation depends on the detector's sensitivity, which is hard to configure accurately and may result in undercounting or overcounting [193]. Overcounting can occur due to excessive sensitivity to adjacent lanes (splashover) or detector interference (chattering) [194]. Furthermore, loop detectors are highly vulnerable to wear and tear [118]. The maintenance costs of induction loops are the highest of different detectors, and their maintenance induces further externalities by preventing use of the road [175]. For this reason, many municipalities are replacing failing loop detectors that have exceeded their lifespans with *non-intrusive* detectors, such as camera and radar detectors [289, 353].

Camera Detectors A camera detector is typically mounted in a fixed position above the roadway, and detection zones are placed on the camera's field of view (Figure 2.3). In the United States, the AutoScope vehicle detection algorithm [248] is used for many cameras. It extracts features to label each detection zone as being in one of three discrete states: "background", "uncertain", and "vehicle". The vehicle counts generated by this algorithm do not reflect actual vehicle volumes — they simply are the number of times each detection zone was actuated. Detection zones must be carefully configured based on their intended purpose. In most cases, they only need to detect vehicle presence or absence. However, if their purpose is accurate counting, the detection zones cannot be large. Otherwise, consecutive vehicles may continuously actuate a detection zone, leading to undercounting. Vehicles that are far from the camera may also be indistinguishable [250].

Inclement environmental conditions also contribute to inaccuracy in camera detector counts. Darkness (due to nighttime or fog) and precipitation (such as rain or snow, which cause glare) obfuscate the visual signal of vehicles, thus making it more difficult for vehicle detection algorithms to isolate them from the background [231, 240, 249]. Even under nominal conditions, shadows can also result in false detections [302]. Finally, detections may be missed on high-speed roads [358].

Radar Detectors Radar detectors, which are also mounted above the road surface, can be used to supplement the camera detectors. They mitigate some of the limitations of camera detectors and require less maintenance [175]. However, they are still affected by inclement environmental conditions, including oscillation caused by wind [242].

2.2.3 Control Hardware and Algorithms

Control hardware is usually installed in a controller cabinet, in which a microprocessor-based *controller* ingests detector inputs and sends control commands to signals. Historically, many controller standards have existed in the United States, but in the early 2000s these were unified by the US Department of Transportation into the Advanced Transportation Controller (ATC) Type 2070 controller standard [296]. Type 2070 controllers are widely used throughout the United States today. In addition to controllers, cabinets typically also include detector processing hardware, load switches that interface between controllers and signals, and a malfunction management unit (or conflict monitor) that performs postprocessing checks as a fail-safe [272].

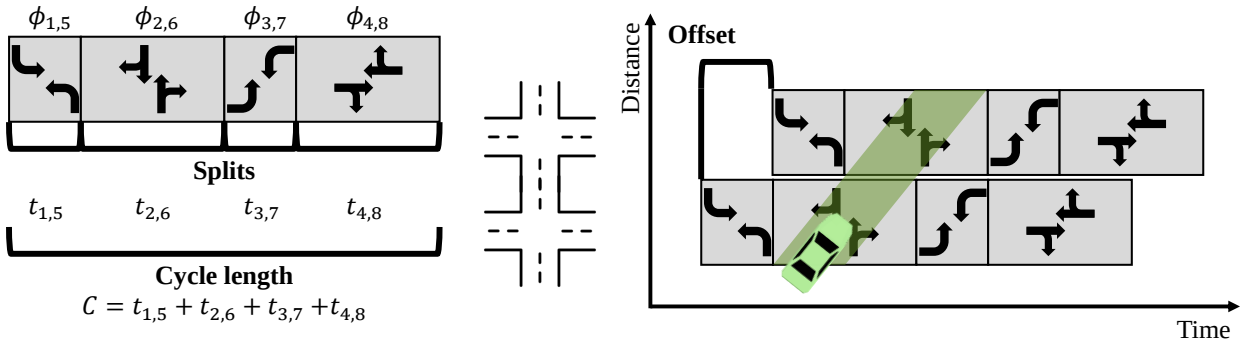


Figure 2.4: Illustration of cycle-offset-split plans for intersections sharing a four-phase signal plan. (L) Diagram of cycle length and splits. (R) Time-space diagram showing effect of offsets. If the offset between two intersections corresponds exactly to a vehicle's free-flow travel time between them, green progression can be achieved.

The most common signal plans implemented by controllers are cyclic, looping through fixed phase sequences in cycles. Under the NTCIP standard, such *cycle-offset-split* (COS) plans are defined by three sets of parameters (Figure 2.4): the overall *cycle length* C, *splits* t_ϕ (proportions of time allocated to each phase), and *offsets* o_ϕ (timing offsets between adjacent intersections to achieve coordination). For offsets, one intersection's controller is usually selected as the master clock. In coordinated control, all intersections in a traffic corridor must share the same cycle length C and splits t_ϕ, with $\sum_\phi t_\phi = C$. The offset must also correspond to the travel time between adjacent intersections at free-flow speed. This has the effect of creating *green progression*, where a vehicle that arrives at the first signal in a corridor on green can continue uninterrupted [182].

Three main types of control algorithms are supported by controllers:

- In *fixed-time control*, a single signal plan or a small number of time-of-day plans are optimised by traffic engineers based on historical traffic volumes. Cycle length optimisation typically depends on setting a desired critical intensity ratio (i.e. the portion of capacity that should be used) for the phase with the highest traffic intensity. The splits, in turn, are selected to balance this ratio across the remaining phases [235]. Software such as Synchro and TRANSYT are used to perform these computations offline [295].
- In *actuated control*, a signal plan is adjusted based on detector inputs through a fixed set of logical rules. An actuated signal plan typically assigns a *minimum green time* to each phase. Each *call* from a detected vehicle lengthens the amount of time available for a phase up to a *maximum green time*, at which point the phase *maxes out*. Alternatively, if no vehicles are detected after an interval known as the *passage time*, the phase *gaps out* [182]. These parameters are, again, usually optimised through offline computation.
- In *adaptive control*, a signal plan is adjusted based on more complex optimization algorithms, such that the mapping from detector inputs to plans can vary over time [97, 122]. These algorithms are usually too complex to be implemented in controllers directly; instead, they are executed in the cloud, and the resulting signal plans are sent to controllers. Software such as SCOOT [349, 348] and Edaptive [95] are used to perform these computations.

Deployed adaptive algorithms largely rely on local search and optimisation techniques to make incremental changes to the cycles, splits, and offsets of predefined time-of-day plans [77, 78], and focus more on leveraging domain-specific heuristics than searching for a globally optimal solution. For instance, many deployed offset optimisation methods are based on the link-pivoting combination method of Day and Bullock [77], which greedily chooses offsets for additional intersections in an ordering based on the road network topology. While these algorithms are designed to minimally disrupt existing workflows for fixed-time control [332], it is not clear that they leverage the full potential of adaptive signal control.

2.2.4 Traffic Simulation

Before an optimised signal plan can be deployed, its quality must be evaluated. However, iterative evaluation and refinement of a signal plan is usually not possible in the real world, because deploying a suboptimal signal plan would result in efficiency and safety costs [110]. *Traffic simulations* provide a safe sandbox within which the performance of signal plans can be assessed and refined. If appropriately constructed based on real-world data, they can serve as digital twins capable of closely capturing the impacts of signal plans on physical traffic.

Traffic simulations exist on a spectrum of how granularly they emulate reality [23, 92]. *Microscopic* simulators provide the most granular simulations, as they simulate the behaviour of individual vehicles (including acceleration, deceleration, and lane changing). This behaviour is generally based on car-following models, which model the acceleration of a vehicle as a time series that depends on its own speed and other vehicles' speeds. *Macroscopic* simulators, by contrast, aggregate vehicles into flows, with time series describing the volume, speed, and density of flows between different points in a road network. *Mesoscopic* simulators provide an intermediate solution between microscopic and macroscopic simulators that balances detail and computational efficiency; they may either model individual vehicles as flows, or organise groups of vehicles into platoons.

What traffic simulators are used in the AI literature and in industry practice? Noaeen et al. [274] reviewed 160 papers in reinforcement learning-based TSC; they found that the most popular simulators are SUMO [10], an open-source simulator; VISSIM, a proprietary simulator; PARAMICS, a proprietary simulator; GLD [408], an open-source simulator; and AIMSUN, a proprietary simulator. Some of these simulators allow mixed microscopic/mesoscopic simulations. All of these simulators aim to be realistic; proprietary simulators generally model more features [310, 376]. Researchers have also created traffic simulators dedicated to training AI methods. On one hand, Zhang et al. [442]'s CityFlow simplifies traffic simulations greatly in exchange for a 20-fold speedup over SUMO; Chapter 4 compares these two simulators. On the other hand, Garg et al. [110]'s Traffic3D simulates environmental perturbations in a detailed 3D environment to increase the robustness of trained agents. Based on my interaction with industry professionals, traffic engineers typically make use of VISSIM; VISUM, its macroscopic counterpart; and SimTraffic, a microscopic simulator that is used by Synchro to perform macroscopic calculations.

What about the traffic simulations that are used in these simulators? Noaeen et al. [274]'s review found that 62% of papers relied exclusively on synthetically-generated simulations, while another 34% of papers included simulations generated based on real-world data. Chapter 5 discusses different methods of generating data-driven traffic simulations.

2.3 Reinforcement Learning

One emerging approach to adaptive TSC is *reinforcement learning* (RL). RL is a paradigm for sequential decision-making wherein agents learn how to act through trial-and-error interactions with an environment. The goal of RL is to learn *policies*, which describe how agents should act conditioned on the current state of the environment.

Early work in RL during the 1980s and 1990s, which included the seminal Q-learning algorithm [394], relied on tabular enumeration of environment states and agent actions. RL remained relatively difficult to scale until the emergence of neural networks and other function approximation methods in the 2010s, which gave rise to *deep RL* [255]. Since then, the popularity and complexity of RL has experienced explosive growth. Game-playing deep RL agents have achieved superhuman performance in card games and video games with high-dimensional state and action spaces and real-time decision-making, such as AlphaGo (Go) [339], Libratus (heads-up no-limit poker) [41], and AlphaStar (StarCraft II) [382]. Deep RL has also found novel applications in practical domains such as robotics, natural language processing, finance, and healthcare [203]. Transportation has been one of the most significant applications of deep RL, with tasks including autonomous driving [174], vehicle dispatching [285] and routing [270], and TSC. I refer the reader to Sutton and Barto [361] for an in-depth review of the history of RL.

2.3.1 Markov Decision Processes

The most common sequential decision-making problem formulation for RL is the *Markov decision process* (MDP), which can be described as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ [359]. In an MDP, a single agent interacts with an environment, which usually consists of a single intersection in the TSC setting, over a number of discrete timesteps $t \in \{0..T\}$. At each timestep t:

States The agent receives a representation of the current environment *state* $s_t \in S$, where S is the set of all possible states. Before the agent has taken any actions, the first state in which it finds itself follows an *initial state distribution* $\rho_0 : \mathcal{S} \to [0, 1]$ [359]. In the TSC setting, the state usually entails an abstracted, numerical representation of a single intersection. As reviewed by Noaeen et al. [274], five of the most common state features in RL for TSC are (1) the queue length per each lane (38%), (2) the current phase (11%), (3) the total vehicle count per each lane (10%), (4) the positions of vehicles (6%), and (5) the speeds of vehicles (6%). A minority of algorithms (3%) directly apply deep image processing techniques to frames from camera detectors or simulators.

Actions Based on s_t, the agent picks an *action* $a_t \in A$, where A is the set of all possible actions.

In the TSC setting, the action is a signalling decision. Noaeen et al. [274] found that, in a majority of algorithms (62%), the action space consists of the index and/or duration of the next phase — which is more myopic than the COS plans discussed in Section 2.2.3. Other algorithms (32%) are based more explicitly on COS plans, with the action space involving the splits (i.e. the length and sequence of phases) or the cycle length.

Transitions The action a_t affects the environment immediately through a probabilistic transition to the next state. This is modelled by the *state transition function* $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Given s_t, a_t, the environment transitions to each possible next state $s_{t+1} \in S$ with probability

$$P(s_t, a_t, s_{t+1}) = \Pr(s_{t+1} \mid s_t, a_t),$$

where $\sum_{s_{t+1} \in \mathcal{S}} P(s_t, a_t, s_{t+1}) = 1, \forall s_t, a_t$. RL methods can be divided into two categories based on how they handle state transitions. (1) *Model-based RL* first learns a model of P before training the agent, usually by fitting it to observed transition probabilities. (2) *Model-free RL* does not learn a model of P; instead, P is considered a notional component of the MDP, and the agent is optimised in expectation over trajectories sampled from P [361].

In the TSC setting, most RL algorithms follow a model-free approach [398]. Given that queue lengths and other common state features are effectively unbounded, the typical sizes of the state and action spaces make the explicit representation and learning of P prohibitively costly.

Rewards After the agent takes action a_t, the environment also gives the agent a numerical *reward* $r_t = R(s_t, a_t) \in \mathbb{R}$, following a *reward function* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. It uses rewards to learn how good taking actions are in various states, so that it learns to take the best action in every state [359].

In the TSC setting, the reward typically denotes the effect of the signalling action on the state (per Noaeen et al. [274], 30% use queue lengths; 6% use vehicle counts), or on vehicle-specific quality metrics (13% use the delays of vehicles, in terms of increase in travel time; 9% use the waiting times of vehicles; 4% use the throughput of intersections).

Policies The goal of the agent is to learn an optimal *policy* $\pi : \mathcal{S} \to \mathcal{A}$ that maps the current state to the action that it should take, $a_t = \pi(s_t)$ (for a deterministic policy), or a distribution over actions, $a_t \sim \pi(a_t \mid s_t)$ (for a stochastic policy). As this is a sequential problem, the agent cannot greedily choose actions to maximise estimated rewards at every timestep, because its actions may have persistent effects (for instance, deciding not to clear a queue in timestep t may lead to congestion in timestep $t + 1$). Therefore, the agent's objective is to optimise its *expected return*

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{s_0 \sim \rho, s_{t+1} \sim P(s_t, \pi(s_t))} \left[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t)) \right] := \mathbb{E}_{s_0 \sim \rho, s_{t+1} \sim P(s_t, \pi(s_t))} \mathcal{R},$$

where $\gamma \in [0, 1]$ is a *discount factor*. The higher the discount factor, the more the agent optimises for rewards in future timesteps; a completely myopic agent has $\gamma = 0$, and an agent that weights rewards from all timesteps equally has $\gamma = 0$ [359].

Over many trajectories, a policy π converges to a discounted *state visitation distribution*, which is given by $\rho_\pi(s) = \sum_{t=0}^{T} \gamma^t \Pr(s_t = s)$; this is not normalised to 1. For a deterministic policy, the state visitation distribution is determined wholly by ρ_0 (the initial state distribution) and π.

Q-Values RL agents need to evaluate the quality of a state, or of an action in some state. These notions are encoded by the *value function* $V(s)$ and the *state-action value function* $Q(s, a)$:

$$V^\pi(s) = \mathbb{E}_{s_0 \sim \rho, s_{t+1} \sim P(s_t, a_t)} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \mid s_0 = s, a_t = \pi(s_t) \right]$$

$$Q^\pi(s, a) = \mathbb{E}_{s_0 \sim \rho, s_{t+1} \sim P(s_t, a_t)} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a, a_t = \pi(s_t), \forall t \in \{1..T\} \right]$$

Intuitively, the value function represents the expected return of beginning in state s_t, and then following π from then on; the Q-function represents the expected return of beginning in state s_t, *taking action* a_t, and then following π from then on [359]. At optimality, the value function and Q-function follow the *Bellman equations*:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s, a, s') V^\pi(s')$$

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') \max_{a'} Q(s', a')$$

RL Algorithms RL algorithms can be broadly divided into two categories.

Q-learning algorithms iteratively learn Q by minimising the loss of the Bellman equation:

$$Q^\pi_{(t)}(s, a) = (1 - \alpha) Q^\pi_{(t-1)}(s, a) + \alpha \left(R(s, a) + \gamma \sum_{s'} P(s, a, s') \max_{a'} Q_{(t-1)}(s', a') \right),$$

where α is a learning rate. Over many iterations, the algorithm converges to a fixed point $Q^* = T(Q^*)$ (where T is an operator for the right hand side of the Bellman equation). Then, the optimal policy is simply given by maximising the learned Q-function, $\pi^*(s) = \text{argmax}_a Q^*(s, a)$ [362].

Policy gradient algorithms explicitly learn a parameterised form for the policy π_θ by optimising its gradient with respect to the Q-value, which is given by the *policy gradient theorem*:

$$J_\theta = \mathbb{E}_{s,a \sim \rho^{\pi_\theta}} Q^{\pi_\theta}(s, a)$$

$$\nabla_\theta J_\theta = \mathbb{E}_{s,a \sim \rho^{\pi_\theta}} \nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)$$

The policy gradient theorem facilitates optimisation by moving the expectation outside the gradient. However, the task of estimating $Q^{\pi_\theta}(s, a)$ remains. In the basic *REINFORCE* algorithm, $Q^{\pi_\theta}(s, a)$ is directly estimated from rewards observed in trajectories. Meanwhile, *actor-critic* algorithms combine policy gradient and Q-learning algorithms by training a *critic* — an estimator of Q — alongside the policy, or the *actor* [360].

2.3.2 Multi-Agent MDPs

The *Markov game* generalises MDPs to the setting of *multi-agent reinforcement learning* (MARL). In MARL, multiple agents exist and interact with each other in an environment; in the TSC setting, this can represent a multi-intersection road network, where each intersection is controlled by one agent [397, 398]. A Markov game has a set of agents $\{1..N\}$, and each agent has an action space $\mathcal{A}_i$ with a *joint action* space $\mathcal{A} = \prod_i \mathcal{A}_i$. Likewise, each agent has its own policy π_i, which can be composed into a *policy profile* $\pi = (\pi_1, \ldots, \pi_N)$. I refer to a policy profile excluding agent i as π_{-i}. At each timestep, each agent i simultaneously executes its own action $a_t^{(i)}$ and receives its own reward $r_t^{(i)} = R_i(s_t, a_t)$. Thus, a Markov game can be described using a tuple $(N, \mathcal{S}, \mathcal{A}, P, R, \gamma)$.

In MARL, the optimal policy for an agent will depend on knowledge of the global state, and of the actions of other agents in the road network. It is possible for all of the policies to be learned in a *centralised* fashion, such that the problem reduces to a single "meta-agent" observing the global state and taking joint actions. They can also be learned in a fully *decentralised* fashion for each agent, where the other agents are viewed as part of the environment. Centralised training can be difficult to scale due to the high dimensionality of the state and action spaces, but decentralised training can result in suboptimal policies that fail to account for each other's behaviour. *Centralised training, decentralised execution* (CTDE) actor-critic algorithms help to mitigate these issues: the critic is centralised but is only used during training, to help coordinate all of the agents, whereas the actors for each agent are decentralised during execution [107, 219].

Coordination can also be introduced as part of the problem formulation. In Markov games where the agents are *cooperative* — as is the case in TSC — a common modification is to share the reward function between all of the agents: $R = R_1 = \ldots = R_N$; V and Q are thus also shared [445]. For such Markov games, *value decomposition* algorithms still train individual Q-functions $\tilde{Q}_i$ to learn each agent's contributions to the overall value, but constrain $Q = \sum_i \tilde{Q}_i$ [294].

In *mixed cooperative-competitive* Markov games, there may be teams of cooperating agents, but not all of the agents share the same reward function [445]. Chapter 7 considers this setting. Here, the agent set is partitioned into a set of disjoint teams $\mathcal{T}_1, \ldots, \mathcal{T}_L \subseteq [N]$, and the environment yields a joint reward for each team $\mathcal{T}_\ell$ according to $R_\ell : \mathcal{S} \times \prod_{i \in \mathcal{T}_\ell} \mathcal{A}_i \to \mathbb{R}$. Each agent maximises its team's total expected return $\mathcal{R}_\ell = \sum_{t=0}^{T} \gamma^t r_\ell^t$. I refer to the joint policy profile for team $\mathcal{T}_\ell$ as $\pi_\ell = (\pi_i, \forall i \in \mathcal{T}_\ell)$, and a joint policy profile excluding all agents in $\mathcal{T}_\ell$ as $\pi_{-\ell} = (\pi_i, \forall i \notin \mathcal{T}_\ell)$.

2.3.3 Partially Observable MDPs

In *partially observable MDPs* (POMDPs), the agent does not directly observe the state s_t. Instead, there is a space of *observations* $\mathcal{O}$. The agent's observations $o_t \in \mathcal{O}$ are assumed to be samples from some state-dependent probability distribution $O(s_t, o_t) = \Pr(o_t \mid s_t)$, with $\sum_{o_t \in \mathcal{O}} O(s_t, o_t) = 1, \forall s_t$. The agent never knows exactly what state it is in; instead, it maintains a probability distribution over states known as a *belief state*, $b_t \in \Delta^{|S|}$, which is based on an initial belief $b_0 \in \Delta^{|S|}$. Thus, a POMDP can be described using a tuple $(\mathcal{S}, \mathcal{A}, P, R, \mathcal{O}, O, b_0, \gamma)$.

Markov games and POMDPs can also be combined into *partially observable Markov games* or *dec-POMDPs*, where there are multiple agents that each have partial observability of the state.

Assuming that the agents are mixed competitive-cooperative, such a problem can be described using a tuple $(N, \mathcal{T}, \mathcal{S}, \mathcal{A}, P, R, \mathcal{O}, O, b_0, \gamma)$. Generally, partial observability is useful for when the same state may be observed differently by agents depending on roles or on randomness. In the TSC setting, there is a global road network state, but not every intersection has access to it; instead, each intersection's agent can only observe the state that is local to that intersection [6, 226].

2.3.4 RL for TSC

TSC has historically been a popular problem for RL practitioners, since it involves scaling up coordination to a relatively large number of agents. RL algorithms for TSC have either focused on improving performance through explicit, centralised coordination, or on improving scalability through implicit, decentralised coordination. This dichotomy existed before the advent of deep learning in 2015, as reviewed by Yau et al. [424]. For instance, Medina and Benekohal [241] learned coordinated Q-functions using the *max-plus* algorithm (a precursor to value decomposition methods), which exchanges messages between agents over a pre-defined *coordination graph*; El-Tantawy et al. [96]'s algorithm involved agents training envisioned policies for their neighbours and best-responding accordingly. Meanwhile, Prabuchandran et al. [283] induced coordination simply by sharing queue length observations between agents.

In 2016, van der Pol and Oliehoek [380] were the first to apply deep RL algorithms to TSC. They used a deep Q-network (DQN) implementation of the max-plus algorithm, with an image-based state representation, and evaluated it on a four-intersection traffic simulation in SUMO. In 2018, Nishi et al. [273] applied *graph neural networks* (GNNs) to RL for TSC; they constructed a graph of traffic volumes for different movements at an intersection, and used a GNN to process this state representation. Out of these works arose an abundance of deep RL algorithms for TSC. Among these, two lines of research are particularly notable.

- Wei, Li, and others developed a series of RL algorithms for TSC from 2019 onwards. Some were explicitly coordinated algorithms, such as CoLight [397], which uses a GNN with attention weights; and CoSLight [308], which uses a Transformer to learn a "collaborator matrix" of probabilistic team compositions.

  Another line of their work focused on learning scalable representations of traffic state, with coordination as a secondary concern. These included IntelliLight, a DQN algorithm that conditions on the phase [400]; LIT, a similar algorithm that uses queue length as a reward [456]; PressLight, a DQN algorithm that uses pressure as a reward [396]; FRAP, an intersection topology-agnostic state representation based on "competition" between phases [455]; and MPLight, a DQN algorithm that shared the parameters of a FRAP network between all agents and used a pressure-based reward [54]. As the culmination of this line of work, MPLight's parameter sharing allowed it to scale up to a 2 510-agent simulation and thus attracted significant attention. I make use of MPLight policies in Chapter 7.

  In parallel, they developed the CityFlow simulator [442] and a version with vehicle behaviour models based on machine learning, CityFlowER [74]; LibSignal, a library that integrates APIs for SUMO, CityFlow, and various benchmark simulation environments [243]; and LibSignal++, a physical test environment with toy vehicles and traffic signals [452].

- Garg, Chli, and Vogiatzis focused less on algorithm development than on the practicalities of deploying RL for TSC. They first introduced the Traffic3D simulator [110]. Then, they designed a policy architecture that directly receives visual input from the simulator and processes it using *convolutional neural networks* (CNNs), which they trained using a REINFORCE-based RL algorithm [108, 109]. After adopting a parameter-shared actor-critic framework [111], they trained the policy using domain randomisation (see Section 2.4.1) and footage from real camera detectors [112]. Finally, they achieved a small-scale physical deployment in Coventry, UK [71].

In addition to LibSignal, Ault and Sharon [14] created RESCO, another benchmark dataset of various traffic simulations and RL-based TSC algorithms. I use this benchmark in Chapters 4 and 7. Alegre et al. [6] implemented an API that wraps around SUMO for RL algorithms; I modified and extended this wrapper for Chapters 6 and 7.

2.4 Deployment Challenges

In Chapter 1, I introduced four practically significant deployment challenges that hinder the deployment of AI technologies in transportation. In the following sections, I review how uncertainty (Section 2.4.1), heterogeneity (Section 2.4.2), assurance (Section 2.4.3), and coordination (Section 2.4.4) arise in the domains of gig driving and TSC.

2.4.1 Uncertainty

One of the central challenges for AI technologies in transportation is that the level of demand and supply is not predictable over time. Transportation systems are not closed [389], as the number of users can vary according to recurrent (e.g. rush hours) and non-recurrent (e.g. weather) factors. Most AI technologies have been evaluated using synthetic datasets, whether road networks or gig requests, that likely fail to capture the full impact of demand and supply variability.

- For TSC, only some, not all, impacts of signalling actions remain consistent under different traffic conditions. Alegre et al. [6] showed that the performance of fixed RL policies degrades in previously unseen contexts, but also has a high variance. While time-of-day plans can handle routine traffic variations, incidents such as lane closures and accidents can lead to inefficiency if not promptly addressed. Yao and Qian [423] showed that traffic incidents can be predicted based on their traffic impacts up to 30 minutes before they are reported.
- For gig driving, the flexibility of driver schedules [56] and the on-demand nature of customer requests [9] introduce volatility into platform dynamics, even if drivers can experientially acquire intuitions and patterns for their work [196]. An additional challenge is imposed by platforms: the destinations of trips are withheld from drivers, which leads to significant difficulties in planning due to not knowing the level of demand at the destination [436]. Some drivers even circumvent the platform to directly communicate with passengers [326].

Putting aside the challenge of predicting future traffic, even observing current traffic level can be difficult for AI technologies without a complete view of the transportation system.

- For TSC, although most RL algorithms assume perfect state observability [274], I showed in Section 2.2.2 that detectors are often inaccurate. What is the impact of this inaccuracy on signal performance measures? I am unaware of any work that has explicitly quantified this, but some indirect results exist. In simulations, Sunkari et al. [358] found that significant differences in delay and queue length arose from different detector types and passage times. Alegre et al. [6] also found that RL performance degrades when the road state is coarsely discretised, resulting in *state aliasing*, where states become indistinguishable to the policy.
- For gig driving, the information asymmetry imposed by gig platforms limits visibility into real-time system conditions for both drivers and third-party tools alike [169]. Even Khan et al. [169]'s tool, which analyses web requests from gig platform apps to infer supply and demand in real time, is constrained by the data that the platforms choose to provide to their users. Tenured drivers have become accustomed to this lack of information and have found ways to adapt, but it still diminishes their sense of control [437].

Some strategies that AI technologies have used to address uncertainty include:

- **Robust training**. Injecting uncertainty directly into the training process allows AI algorithms to adapt to their presence. *Domain randomisation* methods optimise the performance of RL algorithms in expectation over different environmental parameters sampled from a distribution. These parameters can include ambient weather conditions [112] or the probability of vehicle detection [265]. *Transfer learning* and *meta-RL* methods go further: they model uncertainty as tasks sampled from an underlying distribution, and generalise policy parameters across different tasks [275, 265]. *Distributional RL* methods learn all quantiles of the return distribution, and optimise for them at the same time [333]. In solving for driver repositioning strategies that maximise earnings, Chaudhari et al. [52] optimise for worst-case expected earnings given uncertainty in the probabilities of trip origins and destinations.
- **State prediction**. When this uncertainty causes data to not just be noisy but also missing, AI algorithms can be conditioned to behave differently depending on whether data is missing. For instance, Jiang et al. [159]'s TSC algorithm BlindLight predicts the probability of an intersection being "blinded", and uses it to choose between Q-networks for normal and blinded intersections. Alternatively, missing data can be imputed. In the TSC setting, traffic prediction models can be integrated with RL-based control. Mei et al. [244] train models to impute missing states and rewards for RL from neighbouring intersections. External knowledge can also be used: Guo et al. [130] leverage ontological data, while Da et al. [75] use a *large language model* (LLM) to reason about unknown transition dynamics. In the gig driving setting, models exist to accurately perform short-term prediction of surge pricing [27], demand [55], and other information unavailable to drivers.

More fundamental questions exist, however. What uncertainty actually exists in the environment? What part of this uncertainty is useful to model from the perspective of stakeholders? For TSC, most of the work I have cited has arbitrarily defined distributions for noisy or missing data, without much empirical validation. In Chapter 5, I investigate just how much uncertainty in traffic volume arises from detector data. For gig driving, the data that is unavailable to drivers is better defined, but it is unclear how this missing data impacts drivers' decision-making processes. In Chapter 3, I investigate how exposing the presence of uncertainty impacts gig drivers' trust in AI.

2.4.2 Heterogeneity

Variation between individual users can impact their responses to AI technologies in transportation, and the effects of this variation on system-level outcomes must be accounted for.

- For TSC, classes of vehicles such as trucks [123], public transit vehicles [402], and gig drivers' vehicles [138] all have disparate impacts on traffic patterns. Within vehicle classes, factors such as driving speed, reaction time, and braking sharpness affect how individual drivers respond to signals, such as the "dilemma zone" for yellow lights [165]. However, the majority of traffic simulations used for RL training abstract away inter-vehicle variation by treating all vehicles as passenger cars, and many of them ignore pedestrian traffic [274, 442].
- For gig driving, drivers have a variety of motivations and habits. Some rely on it as a primary source of income, while others view it as supplemental income [228, 414]; some drive full-time, while others drive part-time [32, 196, 228]. Accordingly, they also have different responses to the information asymmetry, algorithmic management, and systemic inequity of gig platforms [414, 437]. Although some mechanisms incentivise drivers to report their preferences while maintaining theoretical guarantees [301], it is unknown how well these guarantees hold when accounting for drivers' complex attitudes.

Heterogeneity exists not just in individuals, but also at the level of markets. The performance of AI technologies in transportation can vary across different deployments, especially if they make assumptions about environmental properties that hold in some contexts but not in others.

- For TSC, the most common strategy for RL is to train and evaluate policies on a single road network, often comparing to only a single benchmark algorithm [243, 274]. Additionally, as I discussed in Section 2.2.4, a majority of the simulations used in the literature have been synthetic, not grounded in realistic traffic data [274]. Although the RESCO [14] and Libsignal [243] benchmarks represent useful steps toward homogenising evaluation practices, they have received limited attention outside of their originating research groups.
- For gig driving, it has been consistently recognised that the market entry of platforms has disparate impacts in different cities depending on their socioeconomic makeup [80, 115] and existing transportation infrastructure [115, 329]. Beyond the spatiotemporal locality considered by Ma et al. [223], there has been insufficient consideration of how these dimensions could influence drivers' response to a gig platforms across multiple cities, even though the empirical data of Ong et al. [276] suggests that they do have an impact.

Some strategies that AI technologies have used to address heterogeneity include:

- **Inclusive policies**. Optimising for distinct subgroups of users is one way for AI algorithms to account for their presence. In the TSC setting, various RL algorithms assign different weights to different vehicle classes in reward functions. Some prioritise pedestrians [425, 449], while others prioritise public service vehicles such as buses, ambulances, and firetrucks [112, 188, 315]. In the latter category, some RL methods implement *preemption*, where normal signalling patterns are overridden in the presence of emergency vehicles [330, 352]. In the gig driving setting, Liu and Jiang [214] and Zhou et al. [460] personalise route recommendations based on drivers' features and expressed preferences. Their work is complemented by that of Di et al. [85], who cluster drivers by shift locations and times.

- **Cross-context generalisation**. Incorporating and differentiating heterogeneous environmental contexts into training can allow AI algorithms to adapt to them better. In the TSC setting, generalised state representations such as those of FRAP [455] and MetaLight [433] allow policies to transfer easily between contexts, while Zhang et al. [443]'s meta-RL algorithm groups traffic flows into tasks based on patterns in states, actions, rewards, and travel time. Additional state features for railways [245], trucks [365], and trams [446] also help to capture heterogeneity. In the gig driving setting, Guo et al. [131, 132] predict gig prices by integrating data from gig platforms themselves, taxi services, and public transport; their models thus capture relationships between modes of transportation.

Similar questions as uncertainty arise for heterogeneity. What is the right way to model heterogeneity, and just how much of an impact does it have? When integrating multiple sources of data or multiple objectives into the training process of AI algorithms, how can they be weighted appropriately? In this thesis, I explore the impact of heterogeneity on traffic simulations, upstream of the training process of RL for TSC: Chapter 4 focuses on modelling the heterogeneous behaviour of drivers, while Chapter 5 focuses on integrating heterogeneous detector data.

2.4.3 Assurance

Assurance combines two related notions: *interpretability* and *safety*. First, to ensure that AI-driven decisions are *interpretable* to stakeholders, they must be executed and explained in alignment with their mental models. This is because AI technologies will be applied to established systems in transportation where stakeholders will have acquired mental models through their experience.

- For TSC, the acyclic signal plans common in the RL literature [274] are a far cry from the COS signal plans used by traffic engineers, because they only myopically optimise for the next phase. In Section 2.2.3, I noted that a minority of existing RL methods do optimise cycles, offsets, or splits [162, 230, 334, 419, 434]. However, these methods generally suffer from one of two limitations. Some are heuristic methods that postprocess signalling actions externally to the learning process [162, 419, 434], while others are not scalable because they enumerate the space of valid actions [230, 334].
- For gig driving, the needs and mental models of drivers are a known to be a significant influence on the extent to which they trust platforms' management mechanisms. When drivers' mental models of these mechanisms misalign with reality, they reject platforms' behavioural nudges (such as to surge chase) [50, 196, 436]. While platforms have no incentive to reveal their exact mechanisms, I noted in Section 2.1.3 that drivers engage in sense-making activities [229] that mitigate the impact of this information asymmetry. In doing so, they have developed desiderata for how they want platforms to explain their processes [292, 437].

Second, *safety* constraints must be placed as safeguards against adverse outcomes. Both TSC and gig driving are domains with elevated stakes, where wrong decisions made by AI technologies can have significant impacts on stakeholders. When such impacts do occur, mechanisms for tracing failures back to their root causes must also exist as a means of recourse. There is a trade-off between providing guarantees through enforcing constraints and augmenting performance through loosening constraints; a balance must be carefully struck to satisfy stakeholders.

- For TSC, failing to set appropriate constraints on signal plans can lead to safety issues. Tort cases show that concrete regulatory and mortal risks do arise from improper signalling [368], and thus documentation and justification of signal planning is necessary in case these records need to be subpoenaed in proceedings. Ample scholarship exists regarding how signalling decisions affect crash rates, and these effects have been quantified as *crash modification factors* [225, 345, 412]. However, safety-related objectives are usually incorporated as ad-hoc components of the reward function in RL-based TSC, if at all [99, 121, 209, 215, 426].
- For gig driving, failing to accurately estimate potential demand can lead to opportunity costs in terms of drivers' monetary earnings and time. Guda and Subramanian [129] and Ma et al. [223] have examined opportunity costs in mechanisms that are incurred through mispredictions. However, communicating assurances about potential earnings to drivers remains an open problem. This is because guaranteeing earnings is a difficult task in the presence of uncertainty, even for gig platforms themselves [239]. When platforms provide such guarantees, their mechanisms need to include driver payments to make up for any shortfall [414].

Some strategies that AI technologies have used to address assurance include:

- **Pre-hoc interpretability**. One way to improve interpretability is to ensure that the decision-making processes of AI algorithms are inherently understandable by people. In the TSC setting, RL policies can be designed to balance multiple objectives explicitly in the reward function [46, 121, 439, 440]; to possess monotonicity in state variables [13] and other desirable properties; or even to consist of syntactically correct code, generated through combinatorial search procedures [128]. In the gig driving setting, Zhang et al. [437] and Rao et al. [292] collected suggestions from drivers regarding specific information disclosures and pricing structures that they would like platforms to implement.
- **Post-hoc explainability**. When it is not possible to make the AI algorithm itself interpretable, different explanation methods can be applied to summarise the algorithm's decisions instead. In the TSC setting, many machine learning explainability methods can be applied. *Decision trees*, which branch based on logical rules (see Section 7.2), are more interpretable than neural network-based RL policies [256, 338]; Jayawardana et al. [156] train a decision tree to imitate an RL policy for TSC. *SHAP values*, which measure the contributions of features to model outputs, have also been applied to identify important state features in RL for TSC [161, 323, 441]. Decision trees and SHAP values have also been applied to crash modification factors [405]. In the gig driving setting, Li et al. [202] and Zhang et al. [436] used formative studies to design tools for informing and empowering gig drivers. In particular, Li et al. [202] envisioned a data-sharing network that would quantitatively surface inequalities in task assignment and compensation.
- **Optimisation constraints**. Explicit guardrails on the behaviour of AI algorithms can help guarantee safety. In the TSC setting, unsafe actions can be masked out from selection by RL policies [264]; see Section 6.2.1 for a longer review of safety constraints in RL. Such methods are more rigorous alternatives to approaches based on reward shaping [185]. In the gig driving setting, Raman et al. [291] and Kumar et al. [187]'s income distribution schemes for gig drivers guarantee minimum earnings, although they considered simplified settings.

For assurance, the state of the art of the AI literature in TSC and gig driving have opposite problems. Many interpretability and safety methods have been implemented for TSC, but they do not satisfy the practical desiderata of scalability and alignment with human mental models. I focus on addressing these limitations in RL for TSC by constraining RL policies with practical signalling constraints (Chapter 6), and by improving the scalability of decision tree imitation learning methods (Chapter 7). On the other hand, driver-centred studies have unearthed a variety of desiderata and potential system designs to improve the interpretability of gig driving, but limited progress has been made towards realising these possibilities as deployed systems.

2.4.4 Coordination

To achieve equitable system-level outcomes in transportation, AI technologies must be able to coordinate interactions between many individual agents, all of whom may be self-interested. While various AI algorithms in transportation assume decentralisation between agents, the resulting behaviour lacks guarantees. Specifically, it is not guaranteed that agents will be able to achieve global coordination when they act locally, except when a target state visitation distribution is given [385].

- For TSC, the signalling actions taken by individual intersections are not independent, but will affect the traffic state of upstream and downstream intersections. RL algorithms such as CoLight can emergently learn green progression, and examining the weights of the learned policies show that they capture intuitive patterns of importance along arterials [395, 396]. However, despite their complexity, these RL-based policies are acyclic and still less synchronised than the COS signal plans that are deployed in practice [1], and thus stakeholders cannot reliably guarantee the presence of coordination.
- For gig driving, the choices taken by individual drivers to accept or reject rides will likewise shift the overall distribution of demand and supply. For instance, a route planning tool which recommends many drivers to go to the same, high-earnings location may instead decrease potential earnings by inducing oversupply. Hence, drivers avoid chasing surges [292]. While mechanisms such as those of Garg and Nazerzadeh [113], Ma et al. [223] are designed to be incentive-compatible for individual, utility-maximising drivers, they fail to provide concrete incentives for drivers to coordinate in a way that satisfies demand [276].

Appropriate hierarchical models can help groups of agents to coordinate on achieving higher-level objectives [5, 420]. To this end, both TSC and gig driving have inherent hierarchical structure that can be leveraged by AI technologies, but the practicality of these hierarchies is unclear.

- For TSC, various RL algorithms have used the graph structure of road networks for coordination — either emergently learning hierarchical representations using graph neural networks [273, 386, 397, 435], or explicitly introducing higher-level controller agents that coordinate intersection agents by setting subgoals or reward signals [2, 79, 226, 434, 453]. However, this work has not focused on the practicalities of deploying hierarchical systems. If these computations must take place in the cloud, communication delays will likely exist [222], and therefore appropriate points of synchronisation must be identified.
- For gig driving, platforms can themselves be viewed as controller agents directing the actions of driver agents. Is what is best for a driver also what is best for the platform? Zhang et al.

[438] studied this problem using a cognitive hierarchy approach, and found that individual-level and system-level outcomes are not always in alignment — especially when the strategic sophistications of agents differ. In turn, competitive dynamics between platforms [65, 160] influence their dispatching mechanisms. This breaks the typical assumption for mechanisms that the platform has complete information on supply and demand [223].

Some strategies that AI technologies have used to address coordination include:

- **Mediation mechanisms**. With which other agents should coordinating agents exchange information? The introduction of a mediation mechanism, implicit or explicit, can help agents to better account for others. In the TSC setting, regional *manager agents* can be used to influence the policies of lower-level agents [2, 126, 226]. More implicitly, *coordination graphs* [380, 461] and other collaborator selection mechanisms based on graph attention [308, 331, 397] allow agents to dynamically determine which others to coordinate with. In the gig driving setting, mechanisms have been designed in which a third-party, social welfare-optimising integrator dispatches gigs centrally to individual platforms [22, 393].
- **Communication protocols**. What information should coordinating agents exchange? Standardised communication protocols allow coordinating agents to learn from each other without the need for centralised coordination. In the TSC setting, messages exchanged between agents range from observation embeddings [207, 416, 450] and policy representations [64, 226] to structures that are more explicitly optimised for compactness [38, 458]. In the gig driving setting, d'Orey et al. [90], Yu et al. [427] designed asynchronous, distributed messaging protocols for ridesourcing, in which passengers directly broadcast requests to nearby drivers, and drivers exchange messages to determine which passengers to service.
- **Spatiotemporal partitioning**. When and where should coordinating agents exchange information? In the TSC setting, several RL algorithms have accounted for the fact that not all agents act synchronously, owing to factors such as detection and communication latency. These include algorithms based on distributed computation [413], state prediction [277], and delayed observation sharing [416]. In the gig driving setting, Jin et al. [163] modelled a hierarchy of dispatching agents located in geographically adjacent regions. Other mechanisms have modelled driver repositioning as a spatiotemporally localised problem, leveraging the intuition that coordination is only necessary under periodic peaks in demand [53, 142].

Chapter 3

Missing Pieces

How Do Designs that Expose Uncertainty Longitudinally Impact Trust in AI Decision Aids? An In Situ Study of Gig Drivers

Domain: Gig driving
Challenges: Uncertainty

3.1 Introduction

In this chapter, I study how *uncertainty* impacts people's interactions with AI technologies in transportation, using gig driving as a case study. As I discussed in Section 2.1, gig driving is a complex ecosystem. It does not operate on a strict timetable: drivers are free to drive when they wish, and customers are free to order when they wish. It is also deeply interlinked: drivers and customers all reside in a common pool, and their actions can easily influence each other. This means that the level of demand and supply is never perfectly predictable, even if all of these parties can experientially gain knowledge of common patterns [196, 436].

All of these factors make gig driving an ideal context to study the impact of uncertainty on interactions between AI technologies and their users. In this chapter, I consider a fundamental question that underlies all such interactions. If the goal of AI technologies should be to benefit human stakeholders, what factors would lead a stakeholder to want to use (not want to use) AI technologies as part of their workflows? More specifically, how does uncertainty impact the way people *trust* and *rely* on AI technologies? To answer this question, I focus on *AI decision aids*, which function by (1) recommending decisions and (2) predicting how good the outcomes of following those decisions will be. When uncertainty impacts the predictability of AI decision aids — as is the case when a driver's earnings does not meet expectations created by an AI's predictions — past work has hypothesised that users' trust in the decision aid will be eroded [155, 344].

Prior literature on trust in AI decision aids under uncertainty can be organised into two complementary lines of work. One line of work has studied specific factors that influence trust in AI decision aids, using laboratory experiments in simulated, single-shot, and low-stakes scenarios that

require limited domain expertise [18, 44, 154, 178, 451]. However, context is an important factor for trust in AI [173, 344]. Therefore, the contrived nature of these experiments limits their generalisability to the real-world use contexts of AI decision aids. Another line of work has studied trust in real-world AI decision aids [30, 173, 386], using observational, qualitative studies to assess how existing users interact with these decision aids. These studies are not quantitative assessments of design factors and provide limited insight into how to design new, trustworthy AI decision aids.

Here, I provide a deeper exploration of trust in AI decision aids by combining the strengths of these two lines of work. I contribute the first in situ study of how exposing the uncertainty of an AI decision aid longitudinally impacts users' trust and reliance on the decision aid. Using gig driving as a testbed, I study trust in a real-world, medium-to-high-stakes decision-making scenario where users have existing expertise. Specifically, I comparatively evaluate different designs that expose the potential for misprediction in an AI decision aid. I address the following research questions:

RESEARCH QUESTION 3.1. *How do users' trust and reliance on an AI decision aid depend longitudinally on their perception of its predictive accuracy?*

RESEARCH QUESTION 3.2. *How do different designs that expose the inherent uncertainty in predictive performance impact users' trust and reliance on an AI decision aid?*

I addressed these questions by conducting a longitudinal user study where $n = 51$ gig drivers used a schedule recommendation tool, which did not leverage sophisticated AI methods but was designed to emulate interactions with tools that do. By measuring the trust and reliance of participants over repeated interactions, I tested the effects of exposing uncertainty in the tool's predictions through range-based earnings estimates and hedging text. My quantitative and qualitative findings show that participants' initial perceptions of the tool's accuracy improved their trust in it over time. Range-based uncertainty not only improved trust and reliance in single-shot settings, but also strengthened it over repeated interactions; meanwhile, hedging had the opposite effect.

This chapter was published at the ACM Conference on Fairness, Accountability, and Transparency (FAccT) in 2025 [59].

3.2 Related Work

3.2.1 Trust in AI

Defining and Measuring Trust

Trust is a critical factor that enables individuals and organisations to collaborate productively, especially in uncertain and volatile situations [88]. As automated systems increasingly replace human roles in collaborative tasks, trust has become a central focus of research on human-AI interaction. Empirical studies in various domains have highlighted the role of trust in the acceptance of AI systems by their users [62, 171, 448]. Trust in AI is particularly challenging due to the opacity of many AI models [303]. The literature on AI explainability arose out of these concerns; conversely, AI explainability methods have been empirically found to enhance trust [104, 337].

A lack of uniformity exists in the human-AI interaction literature on how to define and evaluate

trust in AI systems. Kohn et al. [179], Ueno et al. [375] both conducted reviews of trust measures, the latter specifically for human-AI interaction papers. Both reviews found a great diversity of measures and contexts, but also that experimental design choices were seldom informed by or validated against models of trust. I follow Mayer et al. [238] in operationalising and distinguishing the constructs of trust and *reliance*: in the context of AI decision aids, they hypothesise that trust is "the willingness of a person to be vulnerable to the actions of an [AI decision aid], based on the expectation that it will perform a [decision-making task] important to the trustor", and that reliance is the external, behavioural expression of that internal attitude [344].

One point of broad consensus in the literature is that the design of AI systems impacts expectations of their performance, and thus the trust of their users [43, 76, 154, 178, 189, 318, 428]. However, the majority of this work has been based on controlled laboratory experiments. Compared to real-world use contexts of AI decision aids, such experiments have two main limitations.

Trust Over Repeated Interactions

First, past experimental evaluations have largely been *single-shot*, involving only a single session of AI use with no temporal separation between decision points. Ueno et al. [375] reported that most measures of trust were evaluated using questionnaires, and most works evaluated trust only once. However, in the real world, users rarely interact with AI systems only once. As users gain experience with systems, the dynamics of trust between them will also change. Lee and Moray [195] identified three bases of trust that influence the perceived trustworthiness of automated systems by users: *performance* (i.e. perceived task competency), *process* (i.e. perceived transparency of behaviour), and *purpose* (perceived helpfulness and identity of benefactors). Solberg et al. [344] hypothesised that trust in AI decision aids is initially purpose-based; this gives way to performance-based and process-based trust as users experientially develop their perceptions of the AIs. Factors external to the interaction process may also impact trust dynamics, including changes in the environment and in the AI systems themselves (e.g. through model updates [392]).

Most studies that measured trust across multiple interactions have either ignored or attempted to control for learning effects (e.g. through randomisation of scenario ordering) [76, 267, 409]. Several studies have explicitly compared *two* temporally separated trust measurements. Mou and Cohen [261], Mou et al. [262] measured user trust in health e-services during two sessions separated by five weeks; Kunkel et al. [189] compared trust in human-generated and item similarity-based movie recommendations after two recommendations separated by two weeks. However, these studies still provide a limited perspective on the evolution of trust dynamics. Unlike all of these works, I explicitly test the effects of different designs within a more complicated setting, which includes multiple interactions as well as a changing environmental context.

Trust in High-Stakes Situations

Second, past experimental evaluations have largely been *low-stakes*, involving contrived or hypothetical decision-making scenarios in which an element of *risk* and thus vulnerability is largely absent [173]. Under Solberg et al. [344]'s hypothesis, the level of risk in a given context influences the extent to which a user trusts an AI decision aid — even if the user trusts the decision aid in gen-

eral, they may still feel that trusting it in risky scenarios may lead to adverse outcomes. Schoeffer et al. [321], Yurrita et al. [431] provided empirical support for this. Scharowski et al. [318] showed that the influence of risk on trust can be modulated by the design of AI systems; they found that user trust in a loan approval AI system significantly increased in the presence of a "certification label", with a greater increase under a high-stakes context.

In past work, some experiments have studied AI decision aids for decision-making domains entailing high stakes in the real world, such as the medical [43, 44, 30, 154, 386, 342, 415, 422] and financial [282, 321, 318] contexts. However, for practical and ethical reasons, participants in these studies cannot receive feedback from the real world for their decisions. A minority of work has evaluated trust in AI decision aids within their real-world contexts [30, 173, 386, 406], but these have been limited to observational studies that did not compare multiple designs. I contribute a mixed-methods study that assesses the effects of different designs on trust under financial risk, one of the nine different risk domains in a taxonomy introduced by Stuck et al. [351].

Uncertainty and Trust

Uncertainty encapsulates sources of variability that make it difficult for users to reason about the outcomes of relying on an AI, thus increasing the risk of this reliance [351]. AI systems are rarely guaranteed to make perfect decisions, but inherent opacity in their designs makes it difficult for users to ascertain whether good performance can be expected. Various experimental studies have tested designs that improve users' awareness of the presence and impact of uncertainty on AI [18, 49, 178, 268, 391, 418]. This work is grounded in *trust calibration* from the AI explainability literature, which aims to give users realistic expectations regarding when and why AI systems may or may not perform well [34, 372, 451].

Trust calibration allows users to modulate their reliance on AI decision aids, potentially by rejecting their outputs in some contexts [344]. By contrast, *trust enhancement* aims to uniformly improve reliance, which may lead to negative outcomes [451]. An important consideration in trust calibration is whether the imperfection of AI systems arises from *aleatoric* uncertainty (i.e. environmental variability) or *epistemic* uncertainty (i.e. model limitations) [34, 372]. I consider a context where users have existing mental models of how aleatoric uncertainty affects them, but are experiencing the impacts of epistemic uncertainty anew. For both trust calibration and enhancement, prior experiments have lacked real-world context, being limited to hypothetical or simple tasks where participants require little background knowledge.

One thus far underexplored dimension in trust calibration is the use of *lexical hedging*: verbiage that expresses uncertainty. Kim et al. [172] assessed the effects of lexical hedging in a large language model's medical answers on trust but not on reliance. Zhang et al. [447] measured trust and reliance on an AI decision aid with lexical hedging for a contrived shape identification task. I perform a real-world evaluation of two designs for presenting uncertainty in an AI decision aid: presenting scalar ranges for estimates, and qualifying estimates with lexical hedging.

3.2.2 Recommendation Systems

Recommendation systems are mediators of the interfaces between people and the technologies they use. They provide value to their users by mitigating the burden of decision overload and helping them to make better choices [39]; however, they have also benefitted technology providers by driving user engagement and financial opportunities (e.g. [280]). As is the case for AI decision aids, users' innate propensity for trust and domain knowledge also modulate their trust in recommendation systems [176, 47]. While their primary goal is to learn and accurately model the preferences of users, recommendation systems are also persuasive in that their outputs can feed back into the users' preferences [124]. Much work has focused on different longitudinal effects that arise from such feedback loops. On the negative side, recommendation systems can create bias in user ratings [459], concentration in the set of recommended items [105], and degradation in system accuracy [444]. On the positive side, temporal diversity objectives [192] and nudging-based designs [208] can create trust [336] and lead to further exploration of content [208]. However, item preference is ultimately a subjective measure that can vary considerably across users [287]. In this chapter, I explore the feedback loop that arises from a more objective measure, the earnings of gig drivers.

3.3 Pilot Interview Study

To perform an ecologically valid study of trust and reliance in an AI decision aid among gig drivers, I needed to first design a decision aid with practical utility. What kind of AI decision aid would be most relevant to gig drivers? From Section 2.1, one of the biggest challenges faced by gig drivers is the volatility of platform dynamics, which makes it difficult for them to plan their driving activity so as to maximise their profits. I focus on one aspect of planning for gig drivers: choosing *when* to work, subject to their constraints and preferences. There is considerable variation in drivers' habits along this dimension [32, 196, 227]. Choosing *where* to work is another key aspect of planning [436], but I limited my study of uncertainty in this multi-objective problem to a single dimension.

Based on these insights, I concluded that gig drivers would find practical utility in a schedule recommendation tool. Like other AI decision aids [344], such a tool would (1) recommend a set of decisions (i.e. a schedule) to achieve a set objective (maximising their profits), and (2) predict the outcomes (i.e. estimated earnings) of following those decisions. I began my study by creating a prototype of such a tool. Next, I conducted a series of pilot interviews to understand gig drivers' needs and how well the prototype aligned with them. Through this process, I refined the design of the decision aid so that gig drivers would be more likely to find it useful in their daily decision-making. The interview methodology was approved by our Institutional Review Board (IRB).

3.3.1 Prototype Design

Following common practice in UX design [103], I used Figma [106] to create the prototype design. The prototype interface consists of two pages.

First, a **constraint page** (Figure 3.3) elicits scheduling constraints from the user, i.e. when they are available or unavailable during the week on an hourly basis. These constraints would need to

be imposed on any schedule generated by the tool. Unlike Zhang et al. [436], I account for the fact that users' availability may change between days. Additionally, the page also asks users to set a target for themselves — either a maximum number of hours to work, or a minimum amount of money to earn, on a daily basis or on a weekly basis. I considered these as likely goals for drivers to have based on prior driver studies [436, 437]. Due to technical limitations, these questions were implemented as dropdowns, with discretised options used in place of text boxes. However, I consider the impact of this on ecological validity to be minor since I controlled the page.

Second, a **schedule page** (Section 3.4.3) presents the tool's recommended schedule to the user. It was kept static to gather more uniform feedback from participants. The page includes two components. First, it displays a range of estimated weekly earnings, consisting of mean ("On an average week..."), pessimistic ("On a bad week..."), and optimistic ("On a good week...") earnings. Second, it shows a tabular schedule with estimated earnings for each hour of the week, and highlights time slots that the tool recommends for the user to drive during.

3.3.2 Methodology

Participants began by completing a web-based consent form and a demographics survey that collected their age, gender, and education level. After completing the web form, participants were invited via email to complete 20–30-minute audio-recorded Zoom interviews conducted by me. Participants were compensated with a $10 Amazon gift card.

In the first 5–10 minutes, the interview focused on *formatively* understanding drivers' needs and motivations. The questions asked in this portion of the interview were the same as the Intake Survey of the longitudinal user studies (Section 3.5.2). In the last 15–20 minutes, the interview focused on *evaluatively* understanding how well the tool met drivers' needs. To ensure a consistent experience, the Figma prototype was opened in my browser and shown to participants in a screensharing session. First, on the constraint page, the participant was asked to work with me to interact with the page, entering the constraints as if they were using the tool for their actual planning. Then, on the schedule page, the participant was shown a schedule with mocked earnings estimates. Finally, the participant was asked about their overall opinions of the tool.

Transcripts for the interviews were generated by Zoom. I reviewed and corrected these transcripts, then performed structural coding [312]. Afterwards, I and a co-author separately used QualCoder 3.3 [73] to perform open coding [312] and axial coding [313]. We met to reconcile their codes and construct a unified codebook. Finally, I re-applied the updated codes.

3.3.3 Participants

Participants were recruited from the user base of Gridwise, a mobile assistant app for gig drivers, in July 2023. I chose to recruit from this user base to access a relatively large and diverse sample of both historical data and participants. Gridwise distributed recruitment messages to 500 users, but otherwise did not interact with participants. Recipients were sampled from Gridwise users in the United States who had completed at least one gig in a platform linked to the Gridwise app over the week preceding recruitment. I recruited 4 interview participants:

- **P1**: A 39-year-old female with a professional degree who drives exclusively for delivery platforms
- **P2**: A 55-year-old male with less than a high school degree who drives exclusively for delivery platforms
- **P3**: A 29-year-old female with an undergraduate degree who drives more frequently for ridesharing platforms
- **P4**: A 53-year-old male with a professional degree who drives more frequently for delivery platforms

3.3.4 Results

The four pilot interview participants reported a diversity of motivations and routines for driving. While all four participants had specific earning goals, P1, P2, and P4 considered their goals to be important and valued the time flexibility of gig work, whereas P3 was more motivated by the opportunity for human interaction. P1, P2 and P3 had typical times that they drive at; however, P1, P2, and P4 also adjusted their schedules based on demand. All four participants had encountered difficulties in planning due to the unpredictability of demand and/or supply (with P1, P2, and P3 feeling that gig platforms provide insufficient information), and indicated that they would find schedule recommendations to be useful.

All four participants found the initial design of the tool to be generally understandable, and felt that it would be useful for drivers in planning their activity. P1 and P2 liked the fact that the tool presents information to them in a way that reduces the need for guesswork while driving. In particular, P1 suggested that the tool would help mitigate a catch-22: it is not possible to view gig demand information in DoorDash without exiting their Dash (scheduled work period), but doing so seemingly deprioritises them.

On the constraint page, P1 and P4 indicated that the questions aligned well with their goals. P1 and P3 suggested that they would not set the constraints to perfectly align with their routines, so as to receive more information from the tool. On the schedule page, all four participants liked the estimated hourly earnings, with P1, P2, and P3 indicating that they would be helpful in deciding whether or not to work at particular times of day. Yet, P2, P3, and P4 acknowledged that the estimates would only be guesses. P1 and P4 also liked the range of weekly earnings, but P3 felt it assumed they would follow the recommended schedule perfectly. P2 and P3 noted that ranges for hourly earnings would be useful to display.

P1, P2, and P3 all felt that it was better for the tool to have a simple, easy-to-use design. All three indicated that the prototype fulfilled this requirement, although P3 suggested that wording and design improvements would be necessary (in particular, they felt that the monotone colour scheme of the tool was confusing). P2 and P3 felt that the design needed to be mobile-friendly. The participants also mentioned other desiderata:

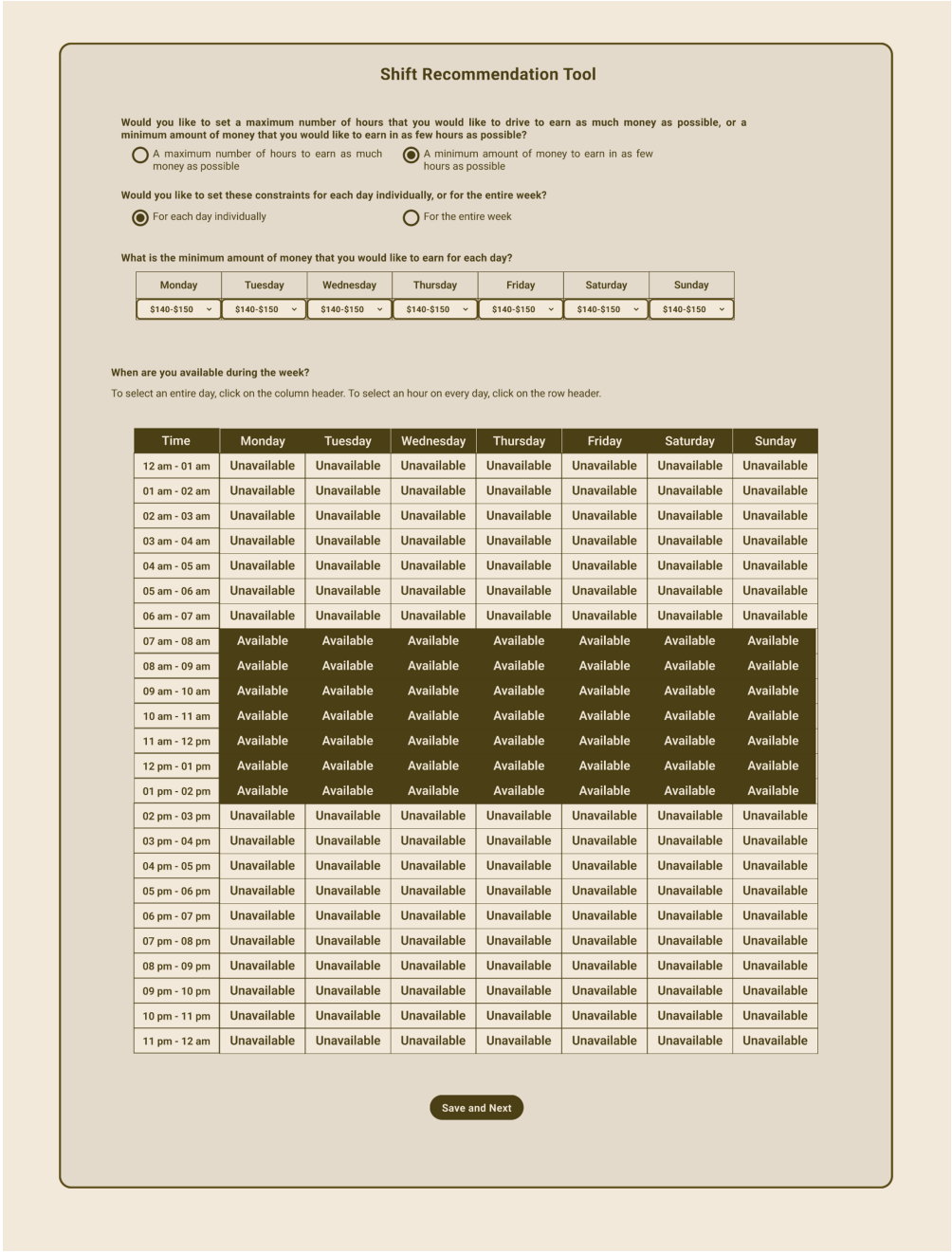
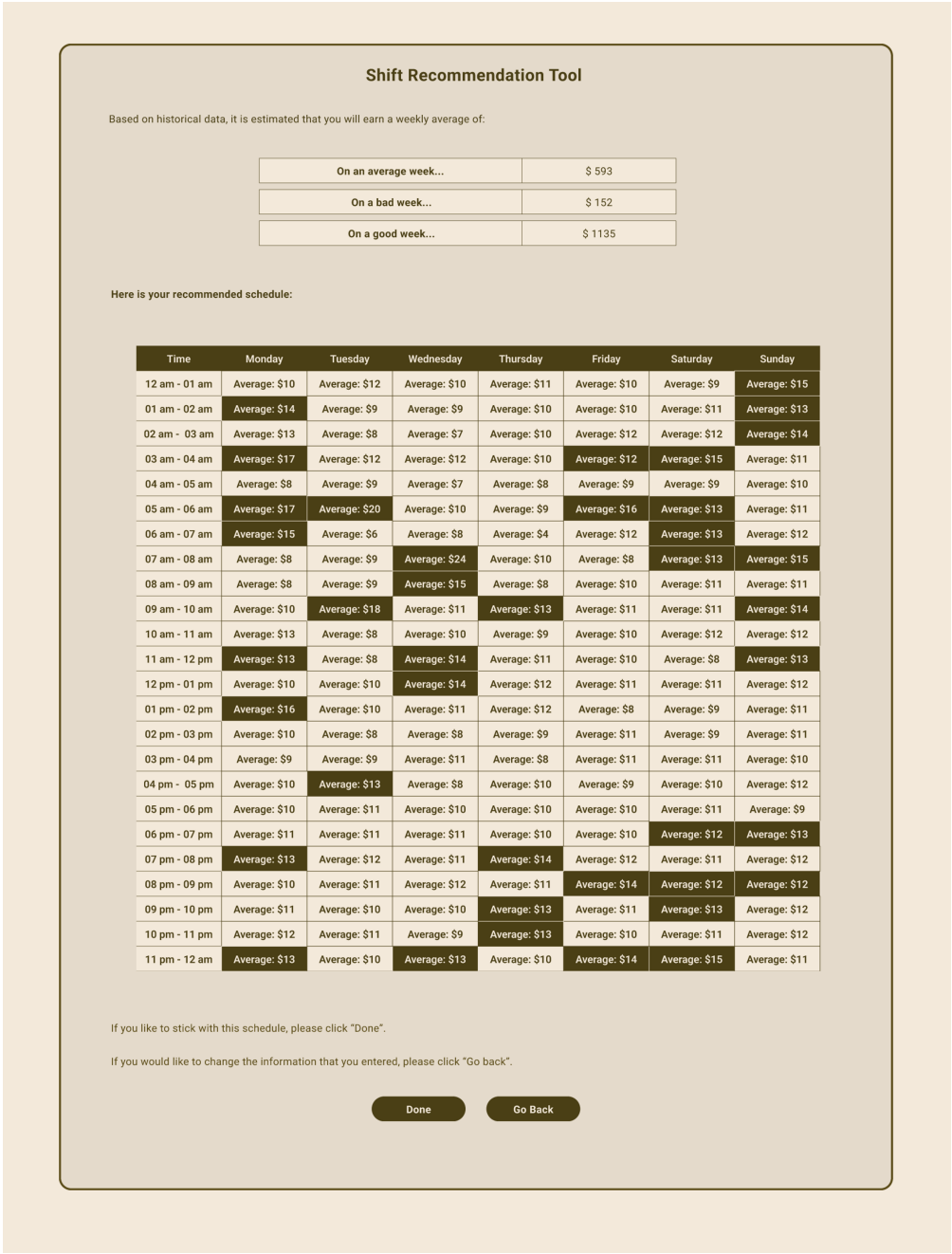


Figure 3.1: Screenshot of Figma prototype for the constraint page. On behalf of participants, I clicked on radio buttons, input fields, and table cells to set their constraints. For technical reasons, input fields for the numerical constraints were implemented as dropdown menus.



Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
12 am - 01 am	Average: $10	Average: $12	Average: $10	Average: $11	Average: $10	Average: $9	Average: $15
01 am - 02 am	Average: $14	Average: $9	Average: $9	Average: $10	Average: $10	Average: $11	Average: $13
02 am - 03 am	Average: $13	Average: $8	Average: $7	Average: $10	Average: $12	Average: $12	Average: $14
03 am - 04 am	Average: $17	Average: $12	Average: $12	Average: $10	Average: $12	Average: $15	Average: $11
04 am - 05 am	Average: $8	Average: $9	Average: $7	Average: $8	Average: $9	Average: $9	Average: $10
05 am - 06 am	Average: $17	Average: $20	Average: $10	Average: $9	Average: $16	Average: $13	Average: $11
06 am - 07 am	Average: $15	Average: $6	Average: $8	Average: $4	Average: $12	Average: $13	Average: $12
07 am - 08 am	Average: $8	Average: $9	Average: $24	Average: $10	Average: $8	Average: $13	Average: $15
08 am - 09 am	Average: $8	Average: $9	Average: $15	Average: $8	Average: $10	Average: $11	Average: $11
09 am - 10 am	Average: $10	Average: $18	Average: $11	Average: $13	Average: $11	Average: $11	Average: $14
10 am - 11 am	Average: $13	Average: $8	Average: $10	Average: $9	Average: $10	Average: $12	Average: $12
11 am - 12 pm	Average: $13	Average: $8	Average: $14	Average: $11	Average: $10	Average: $8	Average: $13
12 pm - 01 pm	Average: $10	Average: $10	Average: $14	Average: $12	Average: $11	Average: $11	Average: $12
01 pm - 02 pm	Average: $16	Average: $10	Average: $11	Average: $12	Average: $8	Average: $9	Average: $11
02 pm - 03 pm	Average: $10	Average: $8	Average: $8	Average: $9	Average: $11	Average: $9	Average: $11
03 pm - 04 pm	Average: $9	Average: $9	Average: $11	Average: $8	Average: $11	Average: $11	Average: $10
04 pm - 05 pm	Average: $10	Average: $13	Average: $8	Average: $10	Average: $9	Average: $10	Average: $12
05 pm - 06 pm	Average: $10	Average: $11	Average: $10	Average: $10	Average: $10	Average: $11	Average: $9
06 pm - 07 pm	Average: $11	Average: $11	Average: $11	Average: $10	Average: $10	Average: $12	Average: $13
07 pm - 08 pm	Average: $13	Average: $12	Average: $11	Average: $14	Average: $12	Average: $11	Average: $12
08 pm - 09 pm	Average: $10	Average: $11	Average: $12	Average: $11	Average: $14	Average: $12	Average: $12
09 pm - 10 pm	Average: $11	Average: $10	Average: $10	Average: $13	Average: $11	Average: $13	Average: $12
10 pm - 11 pm	Average: $12	Average: $11	Average: $9	Average: $13	Average: $10	Average: $11	Average: $12
11 pm - 12 am	Average: $13	Average: $10	Average: $13	Average: $10	Average: $14	Average: $15	Average: $11

Figure 3.2: Screenshot of Figma prototype for the schedule page. A summary of mean, pessimistic, and optimistic weekly earnings is shown at the top of the page, followed by an hourly schedule where recommended cells are highlighted in darker colours. For technical reasons, this was shown as a static page not depending on previously-entered constraints.

- **More granular constraints.** P1, P3, and P4 all suggested ways to limit the scope of the historical gigs used to estimate their earnings. P3 wanted the tool to clarify that the historical data was limited to the city they drive in, and also how recent the data was. P1 and P3 wanted to limit the maximum distance of the historical gigs from their starting point. P4, who works for less popular delivery platforms, wanted to select which platforms the historical gigs came from, and indicated that this would improve their perceived control.
- **Feedback on performance.** P1 and P2 both wanted to compare the tool's estimates with their actual earnings. Regardless of how the estimates compared to reality, P1 suggested that this feedback would be motivating; both P1 and P2 have gamified their gig-driving experiences to compare against either themselves or others. P2 also wanted to compare the estimates with their expenses, and P3 wanted a way to view the overall supply of drivers.

3.4 AI-Based Schedule Recommendation for Gig Driving

In this section, I describe my final design for a AI-based schedule recommendation tool for gig drivers. I use this tool as an exemplary AI decision aid to study the longitudinal relationship between the framing of uncertainty in outcomes and trust.

3.4.1 Decision Aid Design

My tool consists of two modules. First, an **estimation module** prospectively predicts e_{ij}, the earnings that drivers can expect during a specific hour j on a specific weekday i. These could be computed by a machine learning model or averaged from historical data. Second, for each driver, a **scheduling module** uses the estimated earnings and the driver's constraints as inputs to produce an optimal set of working times. To do so, it solves a constrained optimisation problem to set *variables* x_{ij} to 1 or 0, denoting whether the driver is recommended to work in time slot (i, j). Variants for these constraints were retained from the designs I tested in the pilot (Section 3.3.1).

- Some drivers wish to maximise their earnings while minimising their driving hours. For these drivers, the tool maximises the *objective function*: the sum of the estimated earnings e_{ij} for all recommended time slots (i, j) from the estimation module, i.e. $\sum_{i,j} e_{ij}x_{ij}$.

  To set the *constraints*, the tool disallows time slots when the driver is not available, i.e. $x_{ij} \leq a_{ij}$ where a_{ij} is an indicator of whether the driver is available during time slot (i, j). It also places an upper bound on the total hours of recommended time slots per day by b_i, and per week by b_{tot}, i.e. $\sum_j x_{ij} \leq b_i, \forall i; \sum_{i,j} x_{ij} \leq b_{tot}$.
- Some drivers who value earnings to a greater extent set minimum targets for their hourly or daily earnings instead of restricting their driving hours. For these drivers, it minimises the *objective function*: the total hours of recommended time slots throughout the week, i.e. $\sum_{i,j} x_{ij}$.

  To set the *constraints*, the tool disallows time slots when the driver is not available, i.e. $x_{ij} \leq a_{ij}$. It also places a lower bound on the estimated earnings per day by c_i, and per week by c_{tot}, i.e. $\sum_j e_{ij}x_{ij} \geq c_i, \forall i; \sum_{i,j} e_{ij}x_{ij} \geq c_{tot}$.

3.4.2 Interface Design

Next, the front-end interface of the decision aid, based on the prototype from Section 3.3.1, allows drivers to interact with the tool. The interface was implemented as a HTML/CSS/JavaScript website using Django 4.1 [87] and a PostgreSQL database. To mitigate potential biases, I designed my tool to be visually generic and distinct from apps or websites associated with any gig platforms. As with the prototype, the interface consists of two pages.

First, the **constraint page** (Figure 3.3) prompts users to select an optimisation objective: whether to maximise earnings or minimise hours on a daily or weekly basis (b_i, b_{tot}, c_i, and c_{tot}). To maximise perceived control over the tool, I allowed users to choose these options freely rather than assigning them as conditions. The page also elicits hourly availability information (a_{ij}). Compared to the prototype, the constraint page has a more varied colour scheme and clearer instructions, which were improved based on feedback from the pilot (Section 3.3.4).

Second, the **schedule page** (Section 3.4.3) shows the optimal schedule by highlighting the recommended time slots, i.e. the ones that lead to the highest earnings. Again, in the interests of maximising perceived control over the tool, I allowed users to revisit the constraint page until they were satisfied with the schedule. Like the constraint page, the prototype differed from the final design in its colour scheme and clarity of wording.

3.4.3 Interface Conditions

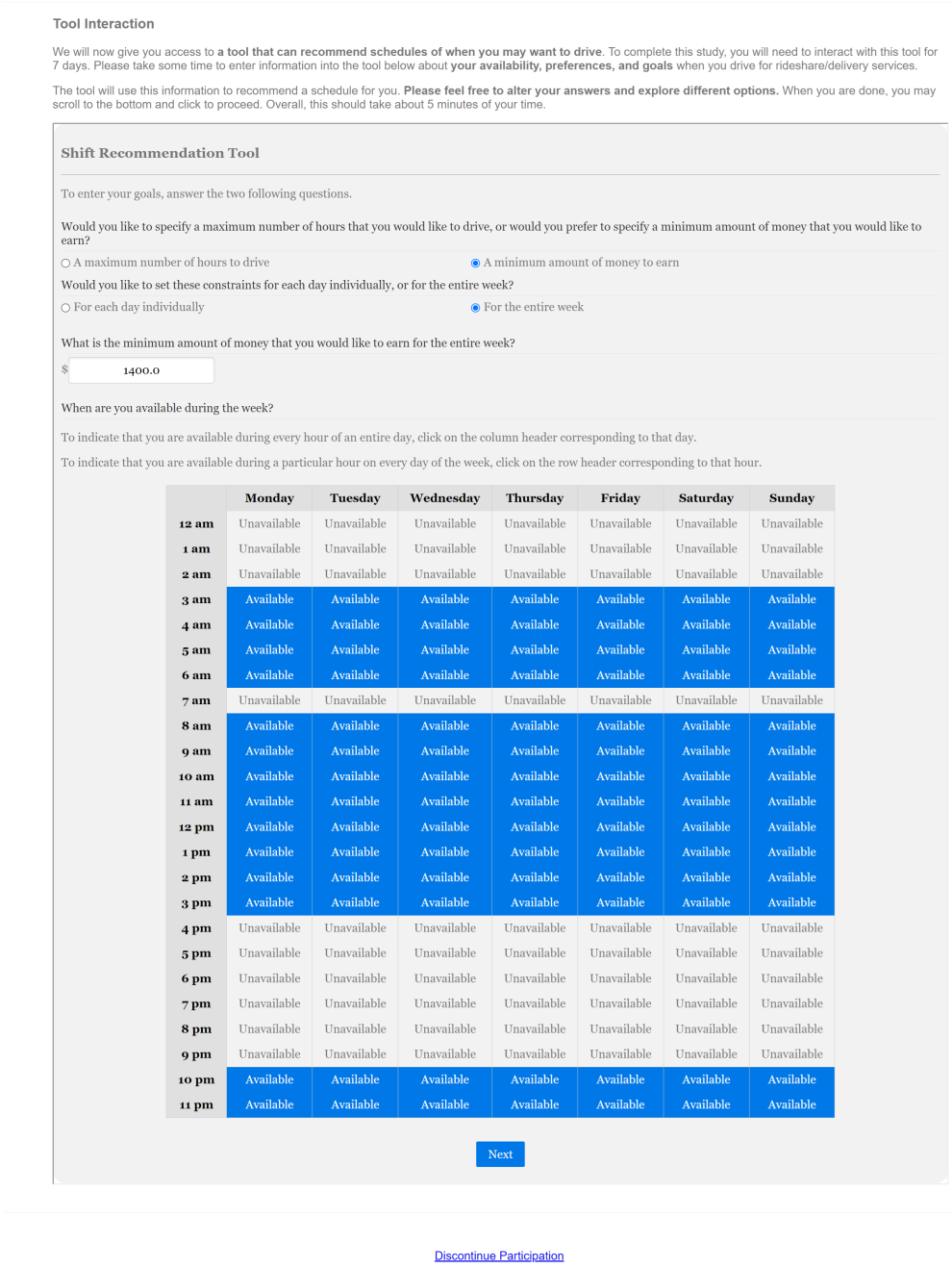
The schedule page uses the outputs of the estimation module to predict how much a driver following the recommended schedule would make per hour and per week. However, uncertainty inherently exists in these predictions, as they are based on historical data, and their realisation is contingent upon which gigs are offered to and accepted by drivers. To address Research Question 3.2, I varied the design of the schedule page between four conditions (Figure 3.4):

(B) **Base condition**. Users were only shown their mean estimated earnings for the week and for each hour in the week.

(D) **Daily estimates**. To assess the effect of introducing additional information irrelevant to uncertainty, users were shown their mean estimated earning for each day instead of their mean estimated weekly earning. As in (B), the schedule still showed mean estimated earnings for each hour.

(R) **Ranged estimates**. To assess the effect of exposing uncertainty through range-based estimates (similar to Prabhudesai et al. [282]), users were shown mean, pessimistic, and optimistic estimates for hourly and weekly earnings. The prototype was most similar to Condition (R). Unlike the final design of Condition (R), however, ranges were not shown for hourly estimates; this was added based on feedback from the pilot (Section 3.3.4).

(RH) **Ranged and hedged estimates**. To assess the effect of exposing uncertainty through lexical hedging (similar to Kim et al. [172] and Zhang et al. [447]), the textual description of the estimates was changed from (R). Instead of "Based on historical data, it is estimated that you will earn", (RH) states "On average, based on historical data, a driver following this schedule will earn".



Tool Interaction

We will now give you access to **a tool that can recommend schedules of when you may want to drive**. To complete this study, you will need to interact with this tool for 7 days. Please take some time to enter information into the tool below about **your availability, preferences, and goals** when you drive for rideshare/delivery services.

The tool will use this information to recommend a schedule for you. **Please feel free to alter your answers and explore different options.** When you are done, you may scroll to the bottom and click to proceed. Overall, this should take about 5 minutes of your time.

Shift Recommendation Tool

To enter your goals, answer the two following questions.

Would you like to specify a maximum number of hours that you would like to drive, or would you prefer to specify a minimum amount of money that you would like to earn?

○ A maximum number of hours to drive ◉ A minimum amount of money to earn

Would you like to set these constraints for each day individually, or for the entire week?

○ For each day individually ◉ For the entire week

What is the minimum amount of money that you would like to earn for the entire week?

$ 1400.0

When are you available during the week?

To indicate that you are available during every hour of an entire day, click on the column header corresponding to that day.

To indicate that you are available during a particular hour on every day of the week, click on the row header corresponding to that hour.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
12 am	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
1 am	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
2 am	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
3 am	Available	Available	Available	Available	Available	Available	Available
4 am	Available	Available	Available	Available	Available	Available	Available
5 am	Available	Available	Available	Available	Available	Available	Available
6 am	Available	Available	Available	Available	Available	Available	Available
7 am	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
8 am	Available	Available	Available	Available	Available	Available	Available
9 am	Available	Available	Available	Available	Available	Available	Available
10 am	Available	Available	Available	Available	Available	Available	Available
11 am	Available	Available	Available	Available	Available	Available	Available
12 pm	Available	Available	Available	Available	Available	Available	Available
1 pm	Available	Available	Available	Available	Available	Available	Available
2 pm	Available	Available	Available	Available	Available	Available	Available
3 pm	Available	Available	Available	Available	Available	Available	Available
4 pm	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
5 pm	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
6 pm	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
7 pm	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
8 pm	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
9 pm	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable	Unavailable
10 pm	Available	Available	Available	Available	Available	Available	Available
11 pm	Available	Available	Available	Available	Available	Available	Available

Next

Discontinue Participation

Figure 3.3: Screenshot of final constraint page, showing constraints entered by interview participant P1.

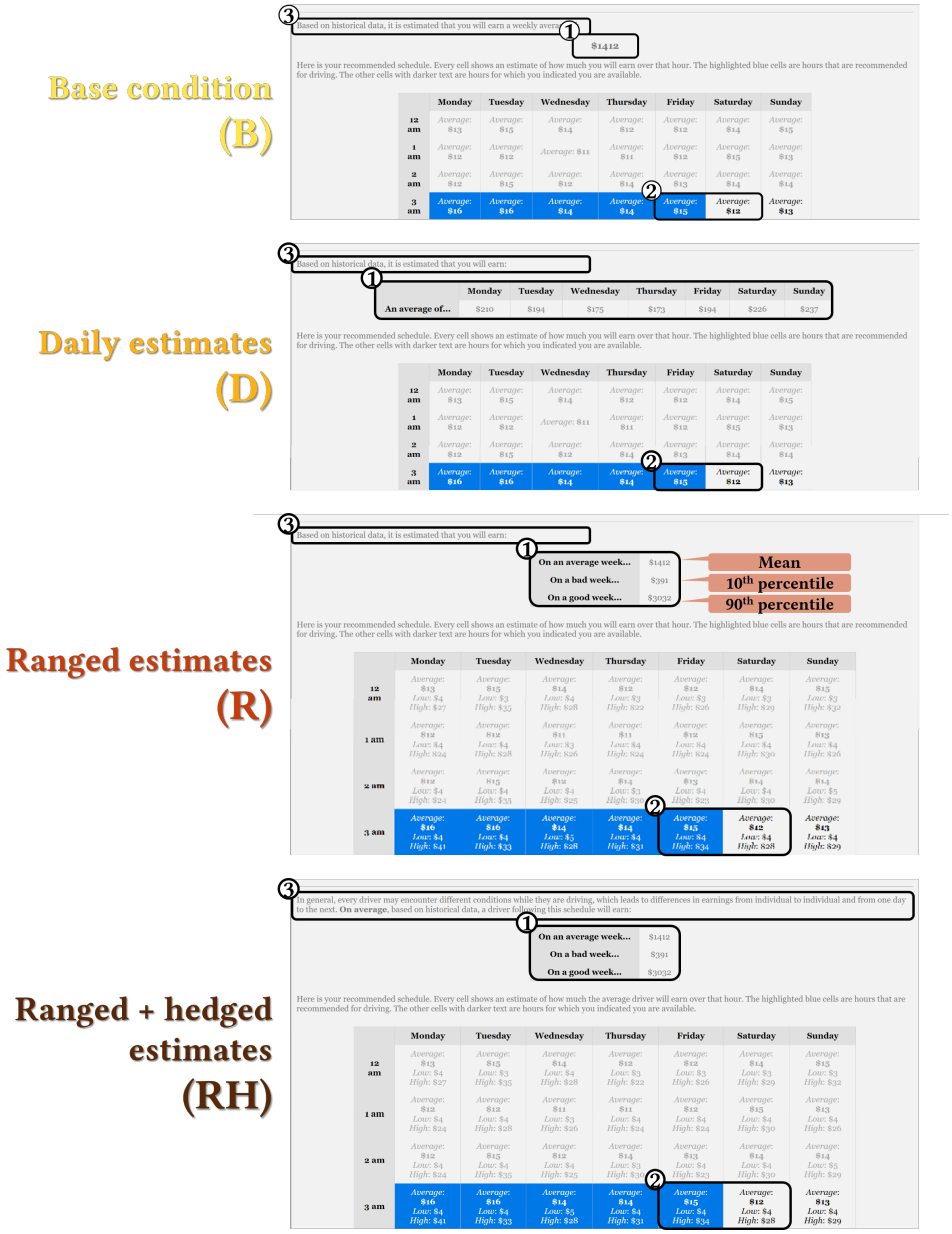


Figure 3.4: Comparison of the four design conditions for the earnings estimates and recommended schedules on the schedule page, with abbreviations following Section 3.4.3. Boxes highlight differences between conditions in three areas: (1) weekly earnings estimates, (2) hourly earnings estimates, and (3) textual description of estimates.

3.5 Longitudinal User Study Design

To address Research Question 3.1, I conducted a longitudinal, in situ user study in which gig drivers repeatedly interacted with my schedule recommendation tool, and I measured their trust and reliance over these interactions. My participants used the tool for 7 days over a 14-day period, with the longer time window meant to accommodate variability in participants' availability. The methodology for this study was approved by our Institutional Review Board (IRB). Figure 3.5 illustrates the flow of the user study; I detail each day's study activities in Section 3.5.2.

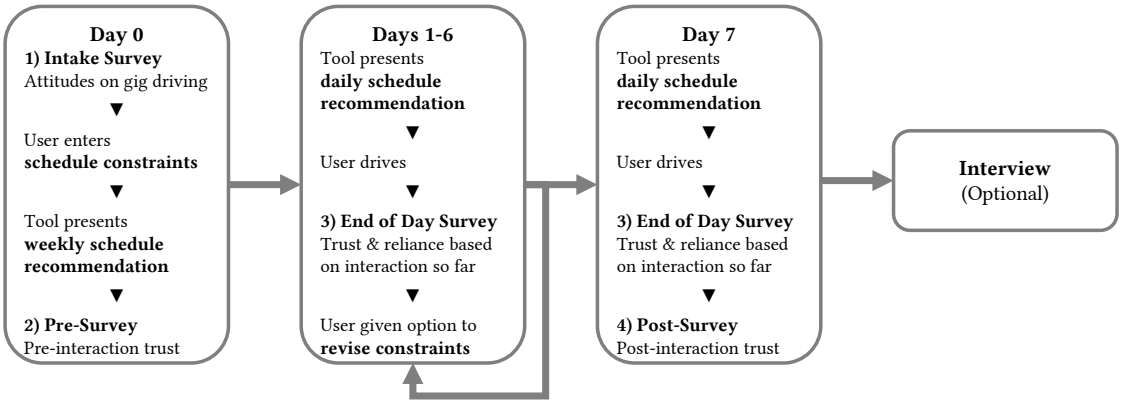


Figure 3.5: Flow of activities for the longitudinal user study. To be compensated, participants needed to complete Day 0 activities.

Based on a pilot conducted with 7 participants in August 2023, I determined that the Intake Survey, Pre-Survey, and tool interaction on Day 0 took an average of 14 minutes and 27 seconds, the End-of-Day Survey took an average of 2 minutes and 23 seconds per day, and the Post-Survey took an average of 2 minutes and 21 seconds. Based on van Berkel and Kostakos [379]'s recommendation of micro-compensation, this led me to set the compensation as an Amazon gift card with $6 for the Day 0 surveys, $2 for each daily survey, and a $20 completion bonus ($40 for full study completion). I made one payment upon study completion or the passage of 14 days.

3.5.1 Participants and Data Sources

As with the pilot interview study (Section 3.3.3), participants were recruited from the user base of Gridwise in September 2023. Gridwise distributed recruitment messages to users who (1) had completed at least one gig in DoorDash, Grubhub, Instacart, Lyft, Uber, or Uber Eats over the month preceding recruitment, and (2) resided in one of the four cities with the historical data used to generate the tool's earnings estimates: Los Angeles, New York, Chicago, and Houston. These were the platforms and cities for which historical data was available.

Accordingly, I generated estimates using gig data from August 2023 in each of these four cities. For each city, the data included approximately 100 000–300 000 gig records distributed evenly across times and weekdays. Hourly earnings were estimated by the mean of what drivers historically earned in this slot, filtered to the participant's city and platforms. I used this static

estimator to focus on the effects of exposing uncertainty in the estimates. Accordingly, I did not emphasise to participants that the schedule page was generated using AI or optimisation, and did not provide any additional information about the data used to generate the estimates.

3.5.2 User Study Activities

Day 0: Pre-Interaction Participants received a link to the study website from a recruitment message distributed by email. After the consent form, they completed the first of four surveys, the **Intake Survey** (Section 3.10.1). This 12-question survey asked about their needs and motivations as gig drivers, along with demographics.

Next, participants were directed to interact with the tool, which I displayed in an iframe to mitigate response bias [82, 177]. They entered their constraints on the constraint page, and received the tool's recommended schedule for the entire week on the schedule page. Participants were assigned to one of the four conditions for the schedule page (Section 3.4.3) uniformly at random, such that each condition had an approximately equal number of participants.

Lastly, participants completed the second of four surveys, the **Pre-Survey** (Section 3.10.1), which was a 5-question survey measuring trust before interaction with the tool (Section 3.6.1).

Days 1–7: Interaction Next, participants began their 7 days of interaction with the tool, beginning on the next day of the week for which they indicated they were available to drive.

On each day, participants first received their recommended schedule for that day, sent via an email scheduled for 30 minutes before the start of their indicated availability. Thus, the tool's outputs were displayed right as they were deciding their driving schedules. During the day, participants independently made decisions about their driving activity; I emphasised that compliance with the recommended schedule was not a condition of full participation.

At the end of each participant's indicated for the day availability, a second scheduled email sent them a link to the **End-of-Day Survey** (Section 3.10.1). This was an 8-question survey that measured their trust in the tool for that day, and their intention to rely on the tool for the next day (Section 3.6.1). If the participant intended to continue relying on the tool, they were then presented with a daily variant of the schedule page. Here, they could review the recommended schedule for the following day, and revise their constraints for the day as desired. Updated schedules were generated by fixing the recommended time slots for previous days using equality constraints and then re-solving the optimisation problem. However, if the participant intended to pause their interaction for one day, an email was sent on the next day, which prompted them to either review the next day's schedule or to pause for an additional day.

Day 7: Post-Interaction On the final day, we removed the last question measuring reliance from the End-of-Day Survey, and added the **Post-Survey** (Section 3.10.1). This was a 10-question survey that retrospectively measured participants' trust and distrust in the tool over the entire user study (Section 3.6.1). After completing the Post-Survey, participants were sent a final email that invited them to participate in an optional **Exit Interview** (Section 3.5.3).

3.5.3 Interview Procedure

For participants who indicated their desire to be interviewed, an audio-recorded Zoom interview of 20–30 minutes was conducted by me. The interview focused on assessing dimensions of participants' experiences that were not evident from the surveys. I began with questions about participants' *motivations and routines*, which led into questions assessing the *constraint page*'s alignment with their decision-making process. Next, I asked participants about the *schedule page*, including how the recommended schedules factored into their decision-making and how it impacted the outcomes of their driving. Further questions focused on the *earnings estimates*, including perceptions of their accuracy and whether participants would've preferred another condition. Then, I asked participants to recall a specific day of interaction in terms of how the tool affected their behaviour for that day and for the following day. Finally, participants were asked for their *overall thoughts* on the tool. The full interview script is shown in Section 3.10.2.

3.6 Quantitative Analysis

Among the 51 participants in the study, 25 (49%) were from Los Angeles, 10 (19.6%) were from New York, 8 (15.7%) were from Chicago, and 8 (15.7%) were from Houston; 4 (7.8%) were aged 18–24, 15 (29.4%) were aged 25–34, 22 (43.13%) were aged 35–44, 8 (15.7%) were aged 45–54, and 2 (3.9%) were aged over 55; 34 (66.7%) were male, 15 (29.4%) were female, and 1 (2%) was non-binary; 5 (9.8%) had a graduate degree, another 16 (31.4%) had an undergraduate degree, another 11 (21.6%) had a professional degree, and another 19 (37.3%) had a high school degree.

Out of these 51 participants, 44 completed at least one day of interaction with the tool, and 34 completed all 7 days of interaction. Starting from Day 0, Day 7 was reached by 6 (46%) of the base Condition (B) participants; 10 (71%) of the daily estimate Condition (D) participants; 7 (58%) of the ranged estimate Condition (R) participants; and 11 (92%) of the ranged and hedged Condition (RH) participants. I show full retention statistics in Figure 3.6.

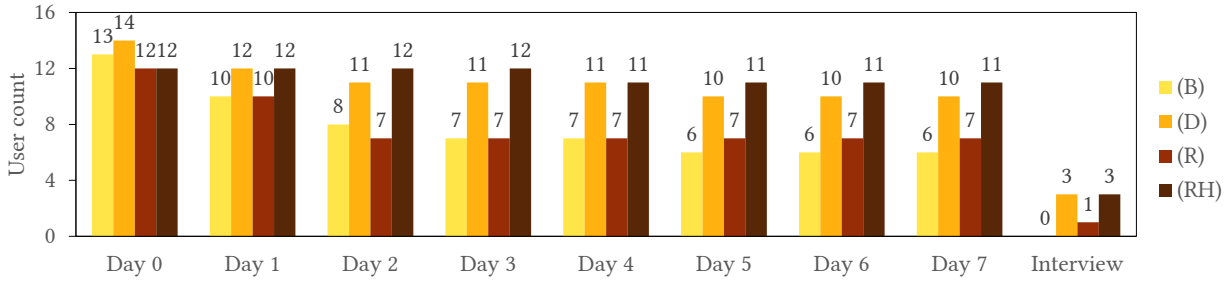


Figure 3.6: Retention statistics for the longitudinal user study, decomposed by design condition. Each day is labelled with the number of participants who completed all study activities for that day. Note the higher retention for Conditions (D) and (RH).

In the following sections, I first describe the metrics that I used to measure the participants' trust and reliance (Section 3.6.1). Then, I analyse my findings from statistical models for these

metrics, specifically those relating to longitudinal effects (Section 3.6.2), and the effects of specific conditions (Section 3.6.3).

3.6.1 Metrics of Trust and Reliance

I measured the **trust** of participants using self-reported measures, following common practice [179]. I used two widely-used instruments for self-reported trust: the Human-Computer Trust Questionnaire (HCT) [233] and the Trust in Automation Scale (TiA) [158]. HCT measures 5 facets of trust using 5 questions each, while TiA measures both trust and distrust with 12 questions. On **Day 0** (pre-interaction), I included 5 items, one taken from each of the HCT's 5 facets of trust, in the Pre-Survey (Section 3.10.1). On **Days 1–7** (during interaction), I included 3 items taken from 3 of the HCT's 5 facets of trust, in the End-of-Day Survey (Section 3.10.1). On **Day 7** (post-interaction), I also included 5 items from the TiA in the Post-Survey (Section 3.10.1), with 3 measuring trust and 2 measuring distrust. I chose not to include questions from this instrument earlier in the user study, as I felt that questions measuring distrust could have biased participants' reliance. All of these questions were presented to participants as 5-point Likert-type scales [67, 66]. From each survey, I computed an overall trust score by first inverting items measuring distrust, if any, and then averaging the Likert-scale responses.

I measured the **reliance** of participants, i.e. the external behavioural expression of trust, using both self-reported measures (End-of-Day Survey, Question 8; Section 3.10.1) and their actual behaviour of discontinuing study participation. Specifically, I computed it as an ordinal variable with three levels: 1, if the participant indicated in the End-of-Day Survey that they intended to rely on the tool *more* tomorrow; 0, if the participant indicated that they intended to rely on the tool *about the same* tomorrow; and -1, if the participant indicated that they intended to rely on the tool *less* tomorrow, or did not complete the next day's study activities.

I use this notation to describe my statistical models:

- `pre_trust_score`: The **Day 0** (Pre-Survey) trust score.
- `trust_score`: The **current day**'s (End-of-Day) trust score.
- `reliance`: The **current day**'s (End-of-Day) reliance score.
- `post_trust_score`: The **Day 7** (Post-Survey) trust score.
- `day`: The day of interaction with the schedule recommendation tool (1–7).
- `user_id`: A randomly-assigned UUID for each participant, used as random effects.
- `condition`: The participant's schedule page condition.
- `estimate_accurate`: A binary indicator of whether the participant perceived their earnings to be about the same as *the tool's estimate* (End-of-Day Survey, Question 4).

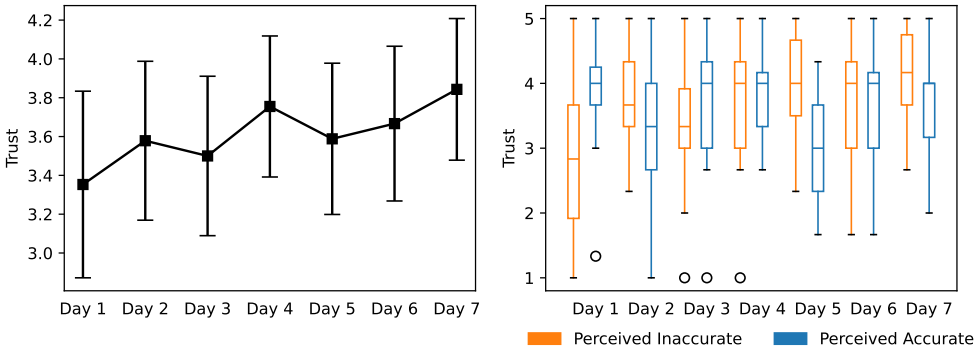


Figure 3.7: (Left) Means and 95% CIs of trust scores for the schedule recommendation tool on Days 1–7 among retained participants. Full statistics are shown in Table 3.3 in Section 3.10.3. (Right) Boxplots of trust scores on Days 1–7, decomposed by perceived accuracy.

3.6.2 RQ1: Longitudinal Effects

Effects on Trust

Participants reported a moderately high level of trust in the schedule recommendation tool ($\mu = 3.631, \sigma^2 = 0.936$). To begin, I analysed how participants' trust in the schedule recommendation tool changed over time. For the 33 retained participants who completed all 7 days of interaction, Figure 3.7 (left) shows an upward trend in the mean trust score. To address Research Question 3.1, I then grouped each day's trust scores based on whether or not participants perceived the tool's estimates as being accurate. Figure 3.7 (right) shows that, on Day 1, perceived accuracy was positively correlated with trust; the interquartile ranges of the trust scores did not overlap between the two groups. This effect was less clear for Days 2–7, where trust scores for the two groups overlapped more extensively.

To further explore the longitudinal effects of perceived accuracy on trust, I fitted a linear mixed model (LMM) for `trust_score` using the R packages lme4 1.1-35.5 [26] and lmerTest 3.1-3 [190]. In this model, these longitudinal effects were modelled by the inclusion of the `day`, perceived accuracy (`estimate_accurate`), and their interaction as independent variables. I also included the `pre_trust_score` to adjust for participants' baseline level of trust in the tool (not on the same scale), and participant IDs as random effects to account for individual variance.

$$\texttt{trust\_score} \sim \texttt{pre\_trust\_score} + \texttt{day} * \texttt{estimate\_accurate} + (1 \mid \texttt{user\_id})$$

My model (Table 3.1) found that participants' pre-interaction trust was significantly and positively correlated with daily trust (`pre_trust_score`: $\beta = 0.471, SE = 0.119, p = 0.00029$). Therefore, **participants' baseline trust persisted throughout their interactions with the tool**. Consistent with Figure 3.7, trust also increased significantly with each passing day (`day`: $\beta = 0.130, SE = 0.027, p < 0.00001$). Also consistent with Figure 3.7, perceived accuracy was significantly and positively correlated with trust (`estimate_accurate`: $\beta = 0.415, SE = 0.167, p = 0.01357$), but it had less of an impact on trust with each passing day of the user study

Factor	Without Condition			With Condition		
	β	*SE*	p	β	*SE*	p
Intercept	1.447	0.468	0.00337**	1.994	0.549	0.00068***
`pre_trust_score`	0.471	0.119	0.00029***	0.411	0.128	0.00256**
`day`	0.130	0.027	¡0.00001***	0.052	0.046	0.25896
`estimate_accurate`	0.415	0.167	0.01357*	0.392	0.167	0.02010*
`day:estimate_accurate`	−0.121	0.037	0.00126**	−0.119	0.037	0.00168**
`condition(D)`				−0.494	0.341	0.15139
`condition(R)`				−0.169	0.360	0.64038
`condition(RH)`				−0.487	0.341	0.15755
`day:condition(D)`				0.111	0.053	0.03548*
`day:condition(R)`				0.103	0.056	0.06976†
`day:condition(RH)`				0.075	0.052	0.14971
Random intercept SD	0.605			0.613		

Table 3.1: Factors and coefficients (β with standard error SE) for my linear mixed model of daily trust scores, without and with the `condition` as an independent variable. Statistically significant coefficients are denoted as † (0.1), * (0.05), ** (0.01), *** (0.001).

(`day:estimate_accurate`: $\beta = -0.121, SE = 0.037, p = 0.00126$). This suggests that, **by the end of the user study, participants' trust was based less explicitly on perceived accuracy**.

Effects on Reliance

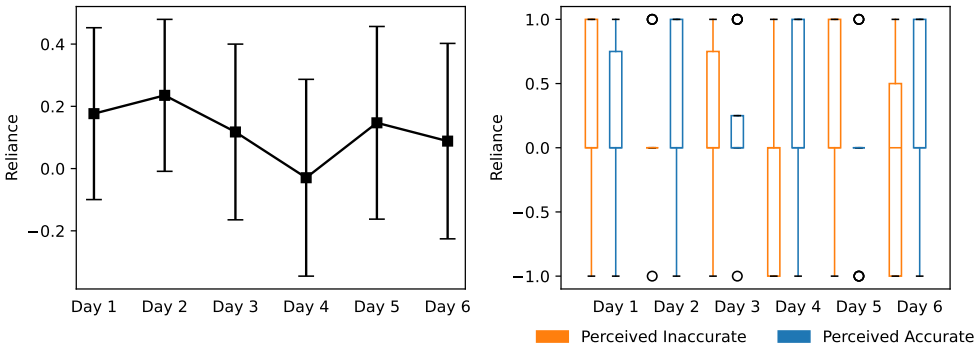


Figure 3.8: (Left) Means and 95% CIs of reliance scores for the schedule recommendation tool on Days 1–6 among retained participants. Full statistics are shown in Table 3.3. (Right) Boxplots of reliance scores on Days 1–6, decomposed by perceived accuracy.

Most participants indicated their desire to maintain their level of reliance on the schedule recommendation tool, corresponding to a reliance score of 0 ($\mu = 0.038, \sigma^2 = 0.643$). Trust and reliance were not strongly correlated ($R^2 = 0.099$); some participants consistently expressed high reliance but also lower trust. The mean reliance score appeared to decrease over time, with the

mean being lowest on Day 4, but I could discern no clear dependence on perceived accuracy (Figure 3.8). To clarify the nature of these longitudinal effects, I fitted another LMM using `lme4` and `lmerTest`. This model was similar to the model for trust, except the `reliance` score was the dependent variable, and I included the `trust_score` as an independent variable:

$$\texttt{reliance} \sim \texttt{pre\_trust\_score} + \texttt{day} * (\texttt{estimate\_accurate} + \texttt{trust\_score}) + (1 \mid \texttt{user\_id})$$

Factor	Without Condition			With Condition		
	β	*SE*	p	β	*SE*	p
Intercept	-0.024	0.422	0.95494	-0.129	0.496	0.79607
`pre_trust_score`	0.104	0.086	0.23316	0.103	0.092	0.27397
`day`	-0.220	0.089	0.01444*	-0.234	0.102	0.02280*
`estimate_accurate`	0.124	0.186	0.50517	0.125	0.187	0.50469
`trust_score`	-0.115	0.099	0.24285	-0.088	0.101	0.38470
`day:estimate_accurate`	0.012	0.048	0.81042	0.008	0.049	0.86786
`day:trust_score`	0.058	0.024	0.01640*	0.045	0.025	0.07127†
`condition(D)`				0.028	0.294	0.92461
`condition(R)`				-0.145	0.308	0.63782
`condition(RH)`				0.139	0.291	0.63414
`day:condition(D)`				0.062	0.067	0.35761
`day:condition(R)`				0.137	0.072	0.05958†
`day:condition(RH)`				0.043	0.065	0.50832
Random intercept SD	0.366			0.382		

Table 3.2: Factors and coefficients (β with standard error SE) for my linear mixed model of daily reliance scores, without and with the `condition` as an independent variable. Statistically significant coefficients are denoted as † (0.1), * (0.05), ** (0.01), *** (0.001).

My model (Table 3.2) did not find significant effects for either the tool's perceived accuracy (`estimate_accurate`) or the `pre_trust_score`. However, two effects were significant: a negative effect from the `day`, supporting my initial observation ($\beta = -0.220, SE = 0.089, p = 0.01444$), and a positive effect from the `day:trust_score` interaction ($\beta = 0.058$, $SE = 0.024, p = 0.01640$). The latter suggests that reliance depended on perceived accuracy indirectly through trust. **Participants who trusted the tool more were more likely to continue relying on it**; this effect strengthened over interactions even as overall reliance weakened.

3.6.3 RQ2: Effects of Conditions

Pre-Interaction Trust

Next, I analysed the effects of the tool's design conditions on trust and reliance, beginning with pre-interaction trust. Conditions (B)/(D)/(R)/(RH) had mean pre-interaction trust scores of 3.338,

3.629, 4.183, and 3.367; Condition (B) had the lowest, and Condition (R) had the highest.

To verify these initial observations, I used the Python package `statsmodels` 0.14.2 [325] to fit an ordinary least squares (OLS) model for pre-interaction trust, with the `condition` as an independent variable. Relative to Condition (B), the daily estimate Condition (D) did not significantly differ in pre-interaction trust (contrast `(D)-(B)`: $\beta = 0.290, SE = 0.369, p = 0.43188$); neither did the ranged and hedged estimate Condition (RH) (contrast `(RH)-(B)`: $\beta = 0.028, SE = 0.384, p = 0.94139$). Yet, Condition (R) had significantly higher pre-interaction trust relative to Condition (B) (contrast `(R)-(B)`: $\beta = 0.845, SE = 0.384, p = 0.02764$) and Condition (RH) (contrast `(RH)-(R)`: $\beta = -0.817, SE = 0.391, p = 0.03685$). Therefore, **exposing uncertainty through range-based estimates initially improved participants' trust**.

Longitudinal Trust and Reliance

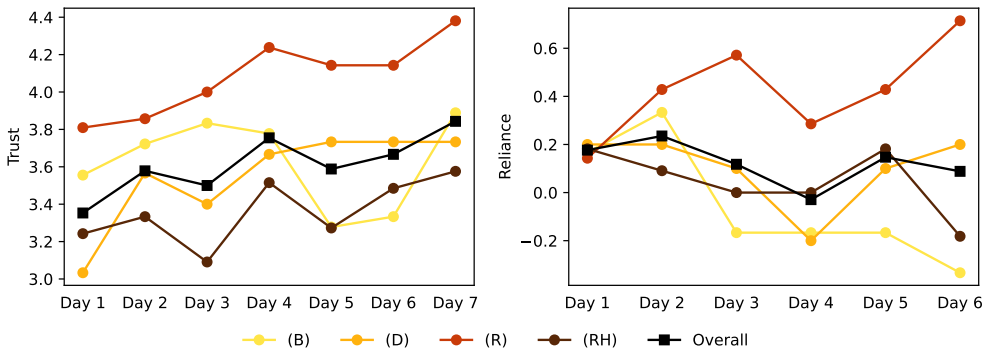


Figure 3.9: Mean trust (Left) and reliance (Right) scores for the schedule recommendation tool among retained participants, decomposed by condition. Note the higher mean scores for Condition (R). Full statistics are shown in Table 3.3 in the appendix.

Lastly, I assessed the longitudinal effects of the schedule page design condition on trust and reliance. In Figure 3.9, I show the mean trust and reliance scores of participants in each of the four conditions. The means of the conditions were in most cases similar to the overall mean, with two exceptions: (1) the mean trust and reliance scores for Condition (R) were the highest of all conditions and showed a generally increasing trend; and (2) the trust scores for Condition (D) were lower on Days 5 and 6. To validate these trends, I added the `condition` and its interaction with the `day` as independent variables to the LMMs that I fitted in Section 3.6.2:

$$\begin{aligned}\texttt{trust\_score} \sim\ & \texttt{pre\_trust\_score} + \texttt{day} * (\texttt{estimate\_accurate} + \texttt{condition}) \\ & + (1 \mid \texttt{user\_id}) \\ \texttt{reliance} \sim\ & \texttt{pre\_trust\_score} + \texttt{day} * (\texttt{estimate\_accurate} + \texttt{trust\_score} \\ & + \texttt{condition}) + (1 \mid \texttt{user\_id})\end{aligned}$$

Condition (D) decomposed estimated earnings on a daily basis, thus providing information irrelevant to uncertainty. My models (Tables 3.1 and 3.2) indicate that this did not significantly improve either trust (`condition=(D)`: $\beta = -0.494, SE = 0.341, p = 0.15139$) or reliance (`condition=(D)`: $\beta = 0.028, SE = 0.294, p = 0.92461$) over the base Condition (B). While my trust model found a significant, positive longitudinal effect in Condition (D) ($\beta = 0.111, SE = 0.053, p = 0.03548$), Figure 3.9 suggests that this does not represent a practically significant trend.

Condition (R) displayed uncertainty in predicted earnings using ranges of pessimistic and optimistic earnings. Again, my models did not find significant marginal effects for Condition (R) over Condition (B) in trust (`condition=(R)`: $\beta = -0.169, SE = 0.360, p = 0.64038$) or reliance (`condition=(R)`: $\beta = -0.145, SE = 0.308, p = 0.63782$). However, Condition (R) had nearly significant longitudinal effects for trust (`day:condition=(R)`: $\beta = 0.103, SE = 0.057, p = 0.06976$) and also reliance (`day:condition=(R)`: $\beta = 0.137, SE = 0.072, p = 0.05958$). This aligns with my observations based on the mean scores in Figure 3.9 as well as my findings for pre-interaction trust (Section 3.6.3). Therefore, despite exposing uncertainty in the tool's earnings estimates, **the ranges of Condition (R) improved participants' initial trust and then led them to maintain their trust and reliance** over daily interactions.

Condition (RH) added lexical hedging to the range-based earnings estimates in Condition (R). This condition was not significantly different from Condition (B) in marginal or longitudinal effects on trust and reliance. On Day 6, participants in Condition (RH) reported significantly lower reliance than participants in Condition (R) (Figure 3.8; $\mu = 0.714, -0.182; 95\%$ CIs $= (0.240, 1.188), (-0.649, 0.286)$), the only such significant pairwise difference on a daily basis (see Table 3.3 in Section 3.10.3). Combined with the pre-interaction trust of Condition (RH) being significantly lower than Condition (R) (Section 3.6.3), I conclude that **the addition of lexical hedging in Condition (RH) reversed the gains in trust and reliance from Condition (R)'s range-based uncertainty**.

Post-Interaction Trust

I fitted an OLS model for the `post_trust_score` with `statsmodels`. This model included the `condition` along with all previous trust (`pre_trust_score` and daily `trust_score`) and `reliance` measurements.

$$\texttt{post\_trust\_score} \sim \texttt{condition} + \texttt{pre\_trust\_score} + \sum_{i=1}^{7} (\texttt{trust\_score\_}i + \texttt{reliance\_}i)$$

For Conditions (B)/(D)/(R)/(RH), the mean post-interaction trust scores were 4.000, 3.720, 3.857, and 3.618. None of these conditions were significantly different from each other, and the coefficients for previous trust and reliance scores were not statistically significant either. The questions I adapted from the TiA asked participants to consider the entire duration of their interaction with the schedule recommendation tool. This broad, retrospective reflection may have failed to capture more nuanced longitudinal changes in trust and reliance like those I described in Section 3.6.

3.7 Qualitative Analysis

Overall, 7 participants completed the exit interview after they completed all seven days of the longitudinal user study. Three of these were from Condition (D), one was from Condition (R), and three were from Condition (RH):

- **P1**: A 47-year-old male with a high school degree driving for Instacart and Lyft in Los Angeles, Condition (D)
- **P2**: A 42-year-old female with a high school degree driving for Uber Eats in Los Angeles, Condition (D)
- **P3**: A 29-year-old female with a graduate degree driving for Lyft in Chicago, Condition (RH)
- **P4**: A 49-year-old male with an undergraduate degree driving for Lyft and Uber in Los Angeles, Condition (RH)
- **P5**: A 46-year-old male with an undergraduate degree driving for DoorDash, GrubHub, and Uber Eats in Chicago, Condition (R)
- **P6**: A 39-year-old male with a graduate degree driving for Lyft, Uber, and Uber Eats in Los Angeles, Condition (D)
- **P7**: A 39-year-old male with a professional degree delivering for DoorDash in Los Angeles by bike, Condition (RH)

To analyse these interviews, I used the same methodology as the pilot (Section 3.3.2). Now, I discuss my findings in relation to participants' motivations for using the tool (Section 3.7.1), perceptions of its accuracy (Section 3.7.2), and perceptions of its uncertainty based on the design conditions (Section 3.7.3).

3.7.1 Motivations and Routines

Participants reported a diversity of motivations and routines for gig driving, which impacted their perceptions of the schedule recommendation tool's usefulness. P1 and P2 viewed gig driving as a primary source of income, and thus found more value in the tool's earnings estimates:

> [The tool was] definitely worthwhile, just because it gave me a number, a projection. [...] They definitely motivate me to keep going the next day. (P1 (D))

Meanwhile, P3–P7 used gig driving to supplement other sources of income, but P4 still viewed his earning goals as important. Unlike other participants, P7 delivered with a bicycle in his spare time. He felt that his current commitment was insufficient to want to use the tool more:

> If I take this job to a full time, take it seriously? I would [want to use it more]. (P7 (RH))

Nevertheless, drivers found value in the tool regardless of their level of motivation. P1–P6 all reported challenges in estimating their potential earnings as a consequence of unpredictability in gig demand, pay, or location, or of gig platforms providing insufficient information. For instance:

> Uber's details that they offer to drivers through their interfaces are sorely lacking. So I'm grateful for the opportunity to interact with this tool. (P6 (D))

3.7.2 RQ1: Perceptions of Accuracy

When evaluating the tool's accuracy, participants weighed its recommendations against their own routines and intuitions. For P1, P3, P6, and P7, the tool was a reference for how well they could perform in their existing routines, rather than something to reshape their routines:

> I still would've followed my routine. [...] I was fortunate enough to at least have the tool make me a schedule based on the routine that I currently do. (P3 (RH))

However, P5 suggested that the tool could use a question-answering approach to nudge users into altering their routines, by first understanding their activity patterns and then suggesting modifications. When the tool was inaccurate, participants reacted in different ways. P1, P2, and P4 observed that instances of the tool being inaccurate decreased their desire to comply with the tool's recommendations:

> If I was making more than what it said,
> I would have done it more consistently on the schedule. (P2 (D))

P4's reactions to inaccuracies were influenced by his expectations. He was motivated on one instance by the tool's estimates exceeding his goals:

> My target's [...] $30 an hour. Because those [estimated earnings] were consistently below $30, [...] I wasn't motivated to study it. But when I saw the 4 to 6 am,
> that kind of piqued my interest. (P4 (RH))

Maintaining consistent perceptions of accuracy over time was important for building trust in this context. P1, P2, and P5 indicated that the outcomes of their first one or two days of interaction impacted their willingness to follow the recommendations for the rest of the study. P4 and P5 indicated that their use of the tool would be strengthened longitudinally if they consistently perceived its predictions as being accurate:

> Once I learned that it was accurate, and I had trust in it, and it was really helping, then I'd probably use it more and more. (P4 (RH))

3.7.3 RQ2: Perceptions of Uncertainty

Some participants recognised that the accuracy of the tool's earnings estimates would be impacted by both their own decisions (P1, P3) and other environmental factors (P6):

> It also depends, too, on the rides that I accept. (P3 (RH))

> It might be true that I might earn the forecasted average earnings.
> But surges can definitely make a difference. (P6 (D))

Note that the tool's uncertainty was not exposed to P6, suggesting that this observation originated from their innate mental model. Recognising the effect of their own agency led P1 and P3, as well as P5, to adopt the tool's estimates as goals for their own earnings:

> [...] setting daily goals of how much money I would like to make [...] was definitely something that I wasn't really doing prior to doing this study or using this tool. (P3 (RH))

Also, P1 suggested that being able to compare their earnings to the tool's estimates in an hourly breakdown would be helpful for goal-setting.

All participants in Conditions (R) and (RH) (P3–P5, P7) appreciated the presence of ranges. P5 compared the tool's range to his own experiences:

> I was always over the average.
> So, to me, I was kind of in my head using that as a low. (P5 (R))

For participants in the other conditions, P1 and P2 indicated that they would've preferred to have had ranges. However, P6 suggested that ranges may lead to disappointment when they are used for goal-setting:

> If they earn less than [the higher number], then they probably might feel disappointed in the tool through no fault of its own, right? If you say $12 to $18, and it comes in at $14, [...] I could understand how folks might look at that as a let down. (P6 (D))

Thus, **ranged-based uncertainty was useful for decision-making, but needed to be calibrated against expectations**.

Both P4 and P5 struggled with the idea that the uncertainty in the tool's estimates could have originated from drivers with habits different to themselves:

> Obviously no one's ever gonna work if it's just $4 or $5 an hour. (P4 (RH))

This was in spite of the lexical hedging presented to P4. At least for this participant, the verbiage in the hedges may thus have failed to achieve its goal of leading him to consider potential sources of uncertainty more carefully.

3.8 Discussion

3.8.1 Key Findings and Implications

Trust in AI decision aids is built both initially and over time. I found that participants' pre-interaction trust in my schedule recommendation tool significantly impacted their trust during interaction (Sections 3.6.2 and 3.6.3). This is consistent with findings in the medical domain that practitioners [43, 45] and patients [261, 262] prefer to gauge their trust prior to interaction. My interviews similarly showed that perceptions of the tool's accuracy in the first two days influenced subsequent trust (Section 3.7.2). Yet, I also found that trust and reliance increased across interactions with the tool. While perceived accuracy had diminishing impacts on trust in later stages of the user study (Section 3.6.2), P4's experience (Section 3.7.2) shows that critical incidents where estimates differ significantly from expectations can cause catastrophic losses of trust [346]. P4 found this difficult to recover from. However, losses of trust could be mitigated prospectively by calibrating perceptions of accuracy, e.g. by emphasising that drivers' outcomes are also a function of their own decisions. This could foster *appropriate reliance* by helping users decide when or when not to rely on the tool [319].

Interactivity could help to maintain trust over time. Losses of trust can also be mitigated retrospectively based on trust repair strategies [278]. Based on my qualitative findings, I hypothesise two mechanisms by which interactivity could help to maintain and repair trust. First, interactivity may enhance perceptions of control. The modes of interaction suggested by my participants, such as hourly breakdowns and question-answering, would assure users that the AI has the intent and agency to capture and learn from their preferences [278]. Second, interactivity could help users

to better recall their experiences and decisions. Until I probed further, most interview participants could not recall whether their earnings significantly differed from the tool's estimates.

The impact of exposing uncertainty on trust in AI decision aids depends on task alignment. Prior work has reached mixed conclusions on how exposing uncertainty in AI impacts trust and reliance. On similar tasks, Zhang et al. [451] found that confidence scores improve reliance, whereas Prabhudesai et al. [282] found that distribution plots dampen trust and reliance. Yang et al. [418] found that the effects of these designs depended on individual characteristics, but my results suggest another dimension: task alignment in the designs themselves. Task-aligned uncertainty representations, i.e. scalar ranges as opposed to distributions, allowed my participants to incorporate uncertainty directly into their decision-making (Section 3.7.3), thus improving trust (Section 3.6.3). This is consistent with findings in the AI explainability literature that domain-aligned explanations are more persuasive [49, 268]. I hypothesise that task alignment also underlies the negative effect of hedging I observed (Section 3.6.3): thinking about other drivers is not helpful when drivers are trying to reason about their own outcomes (Section 3.7.3).

How uncertainty is exposed should be adapted to user subpopulations. My results did not find that a one-size-fits-all approach exists to fostering trust. Even within the same condition, participants exhibited variability in how they reacted to the outcomes of their reliance. Just like in recommendation systems, the same design that sustained trust in one user degraded it in another. Nevertheless, I hypothesise that it may be helpful to adapt uncertainty displays to subgroups within the gig driver population, such as those found by Di et al. [85]. Specifically, my qualitative results point to differing perceptions of accuracy and uncertainty between highly motivated drivers (e.g. P1 and P2) and less motivated drivers (e.g. P7). How can these subgroups be helpful? On one hand, drivers wanted to received estimates from those with habits similar to themselves (e.g. P4 and P5). This aligns with the bandwagon effect from recommendation systems research [210, 357]. On the other hand, drivers would find more value in a personalised system. This aligns with the Barnum effect for recommendation systems [180, 355, 363]. Subgroups that are neither fully personalised nor fully generalised could help to strike a balance between these two cognitive biases. A future large-scale study could help to confirm my hypothesis.

3.8.2 Limitations

My work has two primary limitations. First, I cannot claim that the design of my tool was optimal for engendering trust. My focus was on testing how designs for exposing uncertainty would impact trust and reliance. Thus, I attempted to isolate the effect of this design dimension by refining the tool through a formative pilot study (Section 3.3). Nevertheless, further improvements may have been possible. For instance, I could not provide retrospective breakdowns of participants' earnings due to data availability limitations. Thus, design choices orthogonal to the exposure of uncertainty may have impacted participants' trust and reliance.

Second, despite my best efforts, my sample of drivers was limited. These individuals were at least aware, if not active users, of the Gridwise app, and thus they may have been more focused on their outcomes than the general gig driver population. The trust of users in an AI decision aid is contingent upon their domain knowledge [173, 415], and — as I demonstrate (Section 3.7.1) — the extent to which they integrate the decision aid into their existing routines. A future study aimed at

a broader population of gig drivers could uncover additional insights by explicitly controlling for factors such as full-time status and driver tenure. I was also unable to reach participants who discontinued the user study. Future studies that follow up with such participants would be a valuable source of data on mechanisms of trust loss and repair in longitudinal settings.

3.8.3 Recommendations for Future Work

The paucity of similar longitudinal, in-situ studies in prior work is understandable given the logistical challenges I encountered. I stress the importance of observational studies to improve domain understanding as a basis for longitudinal interventional studies. My pilot interviews helped me to design a tool that was task-aligned, which led participants to find value in it over repeated interactions. Furthermore, my study design aimed to increase perceived control while reducing user burden through flexibility in the scheduling of participation; customisability of the constraints on the AI decision aid; shorter survey instruments; and incremental compensation.

3.9 Conclusion

In this chapter, I considered how uncertainty impacts the deployment of AI technologies in gig driving. I assessed how gig drivers' trust and reliance on an AI decision aid is influenced by designs that expose uncertainty, in the context of an in-situ deployment unlike the laboratory experiments used by previous work. Specifically, I conducted a longitudinal, in situ user study of an AI-based schedule recommendation tool with $n = 51$ gig drivers. Their interactions with my tool impacted their actual earnings. My findings demonstrate that trust can be built by (1) maintaining perceptions of accuracy over repeated interactions and (2) displaying uncertainty in a task-aligned fashion. More generally, they point to the importance of designing with stakeholders in mind as a way to improve the trustworthiness of AI technologies in transportation. While my work in this chapter constitutes an initial foray into improving the deployability of AI technologies for gig drivers, broader challenges surrounding uncertainty, heterogeneity, interpretability, and coordination remain to be addressed (Section 2.4).

3.10 Appendix

3.10.1 Survey Questions

Survey 1 — Intake Survey

1. Please tell us which of the following regions you primarily drive in.
    - Los Angeles
    - New York
    - Chicago
    - Houston

2. Please tell us which of the following services you currently drive for.
    - DoorDash
    - Grubhub
    - Instacart
    - Lyft
    - Uber
    - Uber Eats
    - Other
3. In 2 or 3 sentences, please tell us what you like most about driving for ridesharing and/or delivery services.
4. In 2 or 3 sentences, please tell us what you like least about driving for ridesharing and/or delivery services.

How much do you agree or disagree with the following statements?

5. When I drive, it's important to me that I make some minimum amount of money.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree
6. When I drive, I have an accurate sense of how much money I will make.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree
7. When I don't earn the amount that I expect to from driving, it causes difficulties for me.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree
8. I try to stick to a regular routine for times and places to drive.
    - Strongly disagree
    - Disagree

- Not sure
- Agree
- Strongly agree

9. I am happy with how I currently decide when and where to drive.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree
10. Please tell us your age.
11. Please tell us what gender you identify as.
    - Male
    - Female
    - Non-binary
    - Prefer not to disclose
    - Other
12. Please tell us your highest education level.
    - Less than high school
    - High school
    - Some two-year professional degree
    - Some undergraduate degree
    - Some graduate degree (MS, PhD, JD, or MD)

Survey 2 — Pre-Survey

How much do you agree or disagree with the following statements?

1. I understand how the tool used my answers to generate this recommended schedule.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item U2 from the Perceived Understandability questions of the HCT [233], "I understand how the system will assist me with decisions I have to make." As the constraint page is intended to encapsulate the user's decision-making process, I consider

the generation of the recommended schedule to be how the tool assists the user with their decisions.

2. I feel that I can rely on the tool to produce recommendations which accommodate the things that matter most to me.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item R4 from the Perceived Reliability questions of the HCT [233], "I can rely on the system to function properly." I consider the tool to be properly functioning if its recommendations account for the user's goals and preferences.
3. I feel that the driving times recommended by the tool are as good as what an experienced driver would recommend to me.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item T3 from the Perceived Technical Competence questions of the HCT [233], "The advice the system produces is as good as that which a highly competent person could produce." My tool's advice is its recommended schedule, and to my participants a competent individual would be an experienced driver.
4. I feel that the times suggested by the tool are good even if I don't know for certain that they will maximise my earnings / minimise my hours *[depending on the constraints selected]*.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item F1 from the Faith questions of the HCT [233], "I believe advice from the system even when I don't know for certain that it is correct." Again, my tool's advice is its recommended schedule of driving times. Since no clear notion of correctness applies to continuous estimates of earnings, I reworded this question to focus on alignment with the user's objectives.
5. I would like to use the tool to decide my driving hours in the future.

- Strongly disagree
- Disagree
- Not sure
- Agree
- Strongly agree

This item is based on item P4 from the Personal Attachment questions of the HCT [233], "I like using the system for decision making." I reworded it to better assess participants' level of intended future reliance on the tool.

Survey 3 — End-of-Day Survey

1. How often did you follow the times in the recommended schedule today?
    - I did not follow the recommendations at all
    - I followed the recommendations for one hour during the day
    - I followed the recommendations for two or three hours during the day
    - I followed the recommendations for four or more hours during the day
2. How satisfied do you feel you are with your earnings from today?
    - Very dissatisfied
    - Somewhat dissatisfied
    - Neither satisfied nor dissatisfied
    - Somewhat satisfied
    - Very satisfied
3. As far as you remember, how did your earnings today compare to your expectations?
    - Lower
    - About the same
    - Higher
    - Not sure
4. As far as you remember, how did your earnings today compare to the tool's estimate?
    - Lower
    - About the same
    - Higher
    - Not sure

How much do you agree or disagree with the following statements?

5. I felt that the recommended schedule provided by the tool was easy to follow.
    - Strongly disagree

- Disagree
- Not sure
- Agree
- Strongly agree

This item is based on item U4 of the Perceived Understandability questions of the HCT [233], "It is easy to follow what the system does." Instead of asking the user about the tool's operation generally, I focused the question on the interpretability of its recommended schedule for that day.

6. I felt that the recommended schedule provided all of the information that I needed to decide when to drive.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item R1 of the Perceived Reliability questions of the HCT [233], "The system always provides the advice I require to make my decision." Again, my tool's advice is its recommended schedule. I focused the question on the user's decisions for that particular day.
7. When I was unsure of when to drive today, I followed the recommended schedule.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item F2 from the Faith questions of the HCT [233], "When I am uncertain about a decision I believe the system rather than myself." I focused the question on the user's decisions for that particular day, and reworded "believe" to "follow" to assess compliance more clearly.

    I left out questions based on Perceived Technical Competence and Personal Attachment for length.
8. Which of the following statements do you agree with most?
    - I intend to rely on the tool less tomorrow than I did today
    - I intend to rely on the tool about the same tomorrow as I did today
    - I intend to rely on the tool more tomorrow than I did today
    - I intend to pause my interaction with the tool for one day tomorrow

Survey 4 — Post-Survey

How much do you agree or disagree with the following statements?

1. I feel that I have become familiar with how to use the tool.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item 12 of the TiA [158], "I am familiar with the system." I reworded the question in light of the fact that users did not have any existing experience with using the tool before the study.
2. When I am using a navigation app that suggests routes to me, I feel like I would want to follow the suggestions more if the app asked me questions about my preferences (like this tool did) before giving its suggestions.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is an original question that prompts the participant to consider their interactions with other types of recommendation systems. It assesses the extent to which participants would appreciate granular controls based on their preferences in such systems.
3. Were there questions that you wanted the tool to ask you that it didn't? If so, please tell us about them in 2 or 3 sentences.

How much do you agree or disagree with the following statements?

4. I felt that I was able to trust the schedules recommended by the tool.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item 11 of the TiA [158], "I can trust the system." I focused the scope of this question on the output of the tool, the recommended schedule, rather than the tool as a whole.

5. I felt that I was able to depend on the schedules recommended by the tool for deciding when to drive.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item 9 of the TiA [158], "The system is dependable." Again, I focused the scope of this question on the output of the tool, the recommended schedule.
6. When I am using a navigation app that suggests routes to me, I feel like I would want to follow the suggestions more if the app gave me information about the minimum and maximum possible time of the trip (similar to what this tool did).
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is an original question that prompts the participant to consider their interactions with other types of recommendation systems. It assesses the extent to which participants would appreciate increased exposure of uncertainty through range-based estimates in such systems.
7. I felt that the recommended schedule was misleading.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree
    - Strongly agree

    This item is based on item 1 of the TiA [158], "The system is deceptive." In addition to focusing the scope of this question on the output of the tool, I also reworded "deceptive" to "misleading" to capture the broader possibility of the tool being perceived as unintentionally providing incorrect information.
8. I felt that the recommended schedule harmed my earnings.
    - Strongly disagree
    - Disagree
    - Not sure
    - Agree

- Strongly agree

This item is based on item 5 of the TiA [158], "The system's actions will have a harmful or injurious outcome." In this context, the outcome for the user is their earnings from driving while following the recommended schedule. I reworded the question to assess the outcome retrospectively.

9. If there were any, please identify some of the driving times recommended by the tool that did not align with your expectations.
   - When was the time?
   - In 1 or 2 sentences, why did it not align with your expectations?
10. Do you have any other questions or comments regarding this tool that you would like to share with us?

3.10.2 Interview Scripts

Pilot Interviews

Formative Questions *As the interviews were semi-structured, the script below focuses on the guiding questions that I asked participants. I also probed participants further depending on their responses.*

- Please tell us what you like most about driving for ridesharing and/or delivery services.
- Please tell us what you like least about driving for ridesharing and/or delivery services.
- When you drive, how important is it to you that you make some minimum amount of money daily/weekly?
- When you drive, do you have an accurate sense of how much money you will make?
- Do you try to stick to a regular routine for times and places to drive?
- Are you happy with your current routines in terms of when and where you drive?
- Would getting recommendations for times to drive would be helpful to you?

Evaluative Questions We will show you a tool that can suggest personalised driving schedules. The tool will ask you some questions about your availability and preferences, as well as revenue targets that you might have. Different people might want to use the tool differently. However, we expect a typical user to use it as follows.

First, they would fill in some information about when they are available during the week, along with either how long they want to work or how much they want to make. The tool will then suggest a recommended schedule for the entire week. As they return to the tool every day to plan out their schedules, users will have the opportunity to interact with the tool, tweaking their availability and possible revenue targets to see how the recommendations change.

I begin screensharing the constraint page prototype.

- Here's the initial page of the tool that lets you specify your availability and goals.
- Do you believe you understand what is being shown on this page?
- Do you feel that this tool is asking you the right questions about your availability and goals?
- Are there other important questions that you wish the tool would ask?
- Think about your upcoming week. Using this screen, please tell us what information you think you would want to enter to get a useful recommendation. We will click on the page for you.

I switch to the schedule page prototype.

- Here is an example of a recommended schedule that the tool would generate based on the information you just provided.
- Do you feel that you understand what the recommended schedule is suggesting?
- What part of the recommended schedule do you feel is the most useful?
- What part of the recommended schedule do you feel is the least useful? Is there anything that's missing from the schedule?
- Do you feel the recommended schedule gives you enough information to decide whether you would want to follow it?

Finally, we'd like to ask about your overall opinion of the tool.

- What did you like about this tool?
- What did you dislike about this tool?
- Did you feel that interacting with the tool took too much time, or that it was too complicated or confusing for you? Why or why not?
- What sort of information would increase the chance that you want to use this tool and follow its recommendations? How big of a difference do you think that having this information would make?
- Do you believe that drivers would generally find a tool like this to be useful for when they're planning their driving? Why or why not?

User Study Interviews

Formative Questions Let's start with talking about your driving for rideshare/delivery services in general.

- Could you start by telling us why you are driving?
  - Is it primary or supplemental income?
  - What other commitments do you balance it with (jobs, family, hobbies)?
- To what extent do you rely on making a target amount when you are driving?
- Can you talk through your typical process for deciding when to drive?

Feedback on Constraint Design Now, let's think back to the times when you were interacting with the tool, particularly when it asked you to enter your availability and goals.

- How similar or different were the tool's questions to the way you typically make these decisions?
- Did you feel like you were able to use the tool to adequately specify your main considerations for when you'd like to drive?
  - Were you ever unsure of what information the tool was asking for?
  - Would you have preferred the tool to ask for information differently, or to ask for different information?
- Did you feel like you were able to influence the recommended schedule that the tool generated for you?
  - Did you try to experiment with entering in different information?
- Did you feel that interacting with the tool took too much time, or that it was too complicated or confusing for you?
  - Could you see yourself spending more time interacting with the tool than you did (e.g. to enter more details)? Why or why not?

Feedback on Schedules Now, let's talk about your how the tool's recommended schedules may or may not have influenced your driving activity over the last few days.

- Did you find that the recommended schedules made sense?
  - Did you feel that you understood how the tool used your answers to generate schedules? Why or why not?
- To what extent did you rely on the email reminders of the schedules?
  - Did you ever miss the email reminders?
  - When did you typically check the schedule, if at all?
- If you saw the recommended schedules, how did they impact your process for deciding when to work?
  - To what extent did you follow the schedules?
  - Were there times at which you prioritised your own intuition over the schedules?
  - If so, were there times at which you wished you followed the schedule more closely? Why or why not?
- How did your response to recommended schedules change throughout the week, if at all?
  - Did you look at the schedules more or less as time went on?
  - Were there any particular days on which you wanted to check the schedule more? Why or why not?
- Did you feel that the tool gave you more or different information than you would otherwise get from the services that you drive for/from Gridwise? Why or why not?

- Are there some additional details which could have increased the chance that you followed the recommended schedules?
  - For example, would you have preferred to see the entire week's schedule on every day?
- Did the recommended schedules lead you to drive at different times and/or locations than before?
  - Did this happen early on or later?
  - At what times of day?
- When you followed the recommended schedules, did you feel that you ended up making more money, less money, or about the same relative to before?
  - How closely do you track your earnings in general?
  - Did you track your earnings more closely when using the tool?

Feedback on Estimates

- How much did you focus on the tool's estimates for how much you could earn?
- Did you feel like you could rely on the estimates to achieve your earning goals?
  - Did you feel that these estimates were meant to be accurate projections of how much you could earn, or that they were rough ballpark figures?
- In general, did you feel that the estimated earnings had the right level of detail, or would you have liked to see additional information?
  - *[If participants were in Conditions (B) or (D)]* Would you have preferred to see a range for how much you could earn?
  - *[If participants were in Conditions (R) or (RH)]* Would you have preferred to see a single number for how much you could earn?

I select a particular day on which the participant interacted with the tool. If earnings data was available, this was a day on which the participant earned more than the tool's estimate; otherwise, this was the sixth day of their interaction with the tool.

- Let's talk about [weekday], Day [day] of your interaction with the tool.
- Do you recall the extent to which you looked at the recommended schedule?
  - If you did, do you remember how you decided whether you wanted to follow it? Was this influenced by how much the tool estimated your earnings to be? Why or why not? What did you think of the estimate that the tool gave you?
  - If you didn't, do you happen to remember why? Was this influenced by how much you earned on the previous day? Why or why not?
- Do you recall whether you made more or less than the tool estimated on that day?
  - If there were any differences, do you have any idea why?
- Did that influence your decision to look at the recommended schedule for the next day? Why or why not?

- We checked your records briefly and found that you earned $xxx.xx, compared to the tool's estimate of $xxx.xx. Does that change how you feel at all?

Overall Thoughts Now, we'd like to wrap up with a few general questions about the tool.

- Did you feel that the time you spent interacting with the tool was worthwhile or not worthwhile? Why or why not?
- If you had the option of using a tool like this one, what are the chances that you might actually use it to decide your driving schedule in the future? Why or why not?
- Beyond what you've mentioned already, is there anything else you believe might increase the chance that you would use this tool in the future?
- Do you have any other questions, comments, or concerns?

3.10.3 Full Quantitative Results

In Table 3.3, I report the means, standard errors, and 95% confidence intervals of the daily trust and reliance scores plotted in Figures 3.7 and 3.8.

Day	Condition	Trust			Reliance		
		μ	*SE*	95% CI	μ	*SE*	95% CI
1	Overall	3.353	0.196	(2.872, 3.834)	0.176	0.107	(-0.100, 0.453)
	(B)	3.556	0.444	(2.468, 4.643)	0.167	0.307	(-0.623, 0.957)
	(D)	3.033	0.390	(2.080, 3.986)	0.200	0.200	(-0.314, 0.714)
	(R)	3.810	0.348	(2.959, 4.660)	0.143	0.261	(-0.528, 0.813)
	(RH)	3.242	0.379	(2.315, 4.170)	0.182	0.182	(-0.286, 0.649)
2	Overall	3.578	0.167	(3.169, 3.988)	0.235	0.095	(-0.009, 0.479)
	(B)	3.722	0.416	(2.703, 4.741)	0.333	0.333	(-0.524, 1.190)
	(D)	3.567	0.205	(3.065, 4.069)	0.200	0.200	(-0.314, 0.714)
	(R)	3.857	0.397	(2.885, 4.830)	0.429	0.202	(-0.091, 0.948)
	(RH)	3.333	0.362	(2.447, 4.220)	0.091	0.091	(-0.143, 0.325)
3	Overall	3.500	0.168	(3.089, 3.911)	0.118	0.110	(-0.165, 0.400)
	(B)	3.833	0.331	(3.025, 4.642)	-0.167	0.307	(-0.957, 0.623)
	(D)	3.400	0.364	(2.508, 4.292)	0.100	0.180	(-0.361, 0.561)
	(R)	4.000	0.309	(3.245, 4.755)	0.571	0.202	(0.052, 1.091)
	(RH)	3.091	0.270	(2.430, 3.752)	0.000	0.191	(-0.490, 0.490)
4	Overall	3.755	0.148	(3.392, 4.118)	-0.029	0.123	(-0.346, 0.287)
	(B)	3.778	0.306	(3.028, 4.527)	-0.167	0.307	(-0.957, 0.623)
	(D)	3.667	0.157	(3.282, 4.051)	-0.200	0.200	(-0.714, 0.314)
	(R)	4.238	0.347	(3.390, 5.086)	0.286	0.286	(-0.449, 1.020)
	(RH)	3.515	0.334	(2.697, 4.333)	0.000	0.234	(-0.600, 0.600)
5	Overall	3.588	0.159	(3.199, 3.978)	0.147	0.120	(-0.162, 0.457)
	(B)	3.278	0.475	(2.116, 4.439)	-0.167	0.307	(-0.957, 0.623)
	(D)	3.733	0.257	(3.104, 4.363)	0.100	0.180	(-0.361, 0.561)
	(R)	4.143	0.290	(3.434, 4.852)	0.429	0.297	(-0.336, 1.193)
	(RH)	3.273	0.273	(2.605, 3.940)	0.182	0.226	(-0.400, 0.764)
6	Overall	3.667	0.163	(3.268, 4.065)	0.088	0.122	(-0.226, 0.402)
	(B)	3.333	0.487	(2.142, 4.525)	-0.333	0.333	(-1.190, 0.524)
	(D)	3.733	0.276	(3.059, 4.408)	0.200	0.200	(-0.314, 0.714)
	(R)	4.143	0.340	(3.311, 4.975)	0.714	0.184	(0.240, 1.188)
	(RH)	3.485	0.275	(2.813, 4.157)	-0.182	0.182	(-0.649, 0.286)
7	Overall	3.843	0.149	(3.478, 4.208)	N/A	N/A	N/A
	(B)	3.889	0.351	(3.029, 4.749)	N/A	N/A	N/A
	(D)	3.733	0.247	(3.128, 4.339)	N/A	N/A	N/A
	(R)	4.381	0.286	(3.682, 5.080)	N/A	N/A	N/A
	(RH)	3.576	0.292	(2.862, 4.289)	N/A	N/A	N/A

Table 3.3: Statistics for daily trust and reliance, as measured by End-of-Day Surveys.

Chapter 4

Purpose in the Machine

The Impact of Heterogeneity in Driver Behaviour and Traffic Characteristics on Traffic Simulation Outcomes

Domain: Traffic signal control
Challenges: Heterogeneity

4.1 Introduction

In this chapter, I begin a series of studies of deployment challenges for reinforcement learning (RL) in the domain of traffic signal control (TSC). As I outlined in Section 2.2.4, traffic simulators are an important tool for evaluating the quality of a signal plan.

When these signal plans are based on deep RL, traffic simulators improve the efficiency of training and evaluation in two ways. First, it must be possible to repeatedly evaluate signal plans, as they may need to be iteratively refined based on complex and potentially conflicting requirements from many stakeholders [120]. These include both end-users (i.e. road users) and decision-makers (i.e. traffic engineers and city planners). Second, deep RL algorithms in particular require large quantities of data for training. Real-world traffic studies cannot be continuously-occurring and large-scale due to efficiency and safety costs [110], but traffic simulators can generate large quantities of realistic data for both training and evaluation [442].

At the same time, traffic simulators also allow *heterogeneity* in the deployment contexts of signal plans to be addressed more thoroughly in training and evaluation. First, heterogeneity between human stakeholders can be incorporated: traffic simulators can model different types of road users, who have unique needs and make unique choices that must be addressed individually and equitably. Second, heterogeneity between environmental characteristics can be incorporated: traffic simulators can be produce a diverse set of scenarios that RL algorithms can learn from.

This chapter explores how these two types of heterogeneity are modelled by two different traffic simulators, SUMO and CityFlow, which I introduced in Section 2.2.4. Released in 2001, SUMO [10] is the most popular traffic simulator in RL for TSC [274]. It supports an expansive

framework for simulation definitions, as well as efficient programmatic API access. However, it is single-threaded and thus scales less well to very large road networks. Motivated by this limitation, in 2019 Zhang et al. [442] introduced CityFlow, a traffic simulator designed for applications of RL to traffic signal control. It is multi-threaded, and consequently was reported to achieve a speedup of >20x over SUMO. However, its framework for simulation definitions is more restricted and focuses on aspects which are essential for RL training. Its adoption is limited but increasing [274].

If traffic simulators are to serve as training environments for RL algorithms, the simulators' modelling assumptions must be sufficiently realistic that the resulting signal plans can learn to adapt to a variety of scenarios during deployment. Thus, real-world validation is crucial. However, granular validation with real-world data is usually not possible due to the aforementioned challenges of data collection. Comparisons between simulators take a partial step towards this goal by verifying that different simulators lead to equivalent outcomes. In this chapter, I compare CityFlow against the more granular SUMO. Similar comparisons were conducted by Zhang et al. [442] and Mei et al. [243], but their evaluation used the expected travel time of vehicles — a long-term, system-level outcome — given trained RL policies. Such indirect evaluations do not capture the effects of simulator design and simulation heterogeneity on the *inputs* of RL policies: queue lengths and other instantaneous, low-level features (Section 2.3.1). I performed a more comprehensive comparison to answer the following research questions:

RESEARCH QUESTION 4.1. *Do the low-level simulation outcomes of CityFlow and SUMO have a statistically significant level of distributional equivalence?*

RESEARCH QUESTION 4.2. *How is this distributional equivalence affected by heterogeneity between vehicles in terms of different driver behavioural models?*

RESEARCH QUESTION 4.3. *How is this distributional equivalence affected by heterogeneity in the simulation's traffic characteristics, in terms of traffic demand and road network size?*

This chapter was published at the Winter Simulation Conference (WSC) in 2023 [57].

4.2 Related Work

4.2.1 Validating Traffic Simulators

Validation is an important yet challenging aspect of the development of traffic simulators that ensures their fidelity to the real world. A multitude of road networks have been used to validate SUMO itself [10, 29] and to calibrate its car-following models [184] in comparison to detector data. For instance, Lobo et al. [218] compared traces for vehicle counts and crossing times between SUMO and detector data for a road network in Ingolstadt, Germany; their simulation is used in this work. Meanwhile, CityFlow was validated by comparing average travel times under various traffic volumes to that of SUMO [243, 442].

A number of studies have compared outcomes from multiple simulators; my work in this chapter falls in this setting. Maciejewski [232] compared vehicle counts under different traffic demand and driver behaviour settings for SUMO, the commercial simulator VISSIM, and TRANSIMS.

Several studies involved SUMO and the commercial simulator AIMSUN. Leksono and Andriyana [199] compared travel times and queue lengths in these simulators under different traffic management interventions for a roundabout in Norrköping, Sweden. Ronaldo and Ismail [305] used t-tests to compare flows and speeds in these simulators to observations from a highway interchange in Stockholm, Sweden. Baza-Solares et al. [28] applied t-tests to vehicle counts from the two simulators for a road network from Bucaramanga, Colombia. This work differs from prior approaches in that: (1) the measures used relate to the distribution of outcomes across the network, not just at individual points; and (2) these measures were evaluated across multiple road network scales.

4.2.2 Modelling Driver Behaviour

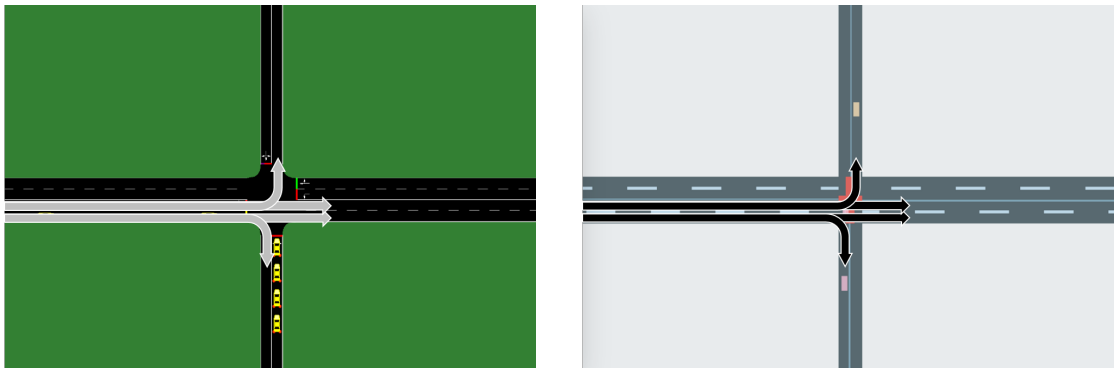
There has been a significant body of literature on building realistic models of driver behaviour. Car-following behaviour was the first to be modelled, with the early Gazis-Herman-Rothery model being over 50 years old. Subsequent work has yielded optimal velocity, fuzzy logic, collision avoidance, action point, and cellular automaton models [204, 311]. I implemented collision avoidance and action point models. Lane-changing behaviour has been modelled by a newer, separate line of work [258, 457], which has produced models based on rules, discrete choice, game theory, and cellular automata. Some work has built unified models of car-following, lane-changing, and other driver behaviour [236, 371]. Here, car-following and lane-changing models were considered separately but co-varied in experiments to elucidate the effects of their interactions. The most relevant prior work is Capela Dias et al. [48], who analysed the impact of driver behaviour on system-level travel time in SUMO; by contrast, I focus on distributional equivalence between two simulators in terms of lower-level outcomes.

4.3 Comparing SUMO and CityFlow

Both CityFlow and SUMO are microscopic traffic simulators. They represent road networks as graphs, with intersections as nodes ("junctions" in SUMO) and roads as edges between nodes [20]. Vehicles are generated by flows; all vehicles within a flow share similar attributes, including behavioural parameters and routes [19]. Routes are defined as sequences of edges. In order to follow its route, a vehicle will proceed down a road edge until it arrives at an intersection. As shown in Figure 4.1, permissible movements through an intersection — through, right, and left — are defined by specific pairs of lanes ("roadlinks" in CityFlow, "connections" in SUMO).

When executing the simulation loop, both simulators perform the same procedures in each timestep, although the two simulators perform them in different orders (Figure 4.2):

- **Routing**. Vehicles decide whether to keep or alter their current routes.
- **Traffic signal logic**. The states of signals are updated according to signal plans.
- **Car-following**. Vehicles decide how to follow vehicles in front of them.
- **Intersection navigation**. Vehicles decide how to traverse roadlinks, if necessary.
- **Lane-changing**. Vehicles decide whether and how to change lanes.



(a) SUMO (b) CityFlow

Figure 4.1: Screenshot of an intersection in the arterial4x4 road network in SUMO and CityFlow, showing connections in SUMO/roadlinks in CityFlow.

- **Environment update**. After all of these decisions have been made, all of the simulation objects are updated with their current state.

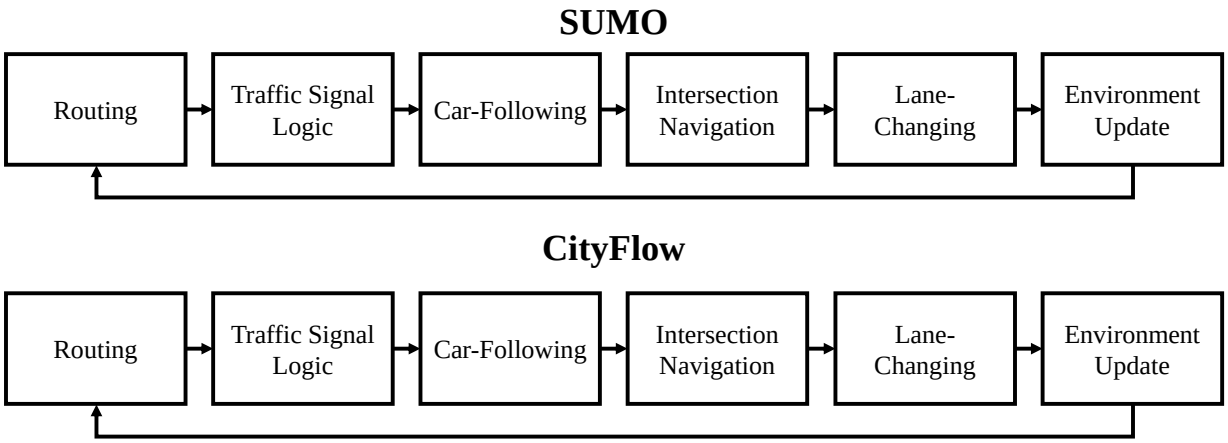


Figure 4.2: Diagram of simulation loop for SUMO and CityFlow.

However, as shown in Table 4.1, key differences between the two simulators exist in finer details of implementation [243]. Additional differences exist in how these simulators model heterogeneity in vehicle behaviour; I discuss these aspects in Section 4.4.

Where possible, I designed the experiments to control or co-vary differences between the simulators. In particular, I converted flows in both CityFlow and SUMO to be single-vehicle and deterministic, and to share the same routes. However, differences remain, particularly aspects of driver behaviour (Section 4.4) driven by randomness in SUMO; by contrast, CityFlow only uses random generation for vehicle priority. Such uncontrolled factors may account for their differences.

Feature	SUMO	CityFlow
Performance	Single-threaded if using `libsumo`, relatively slow due to socket communication if using `traci`	Multithreaded, relatively fast
Vehicles	Heterogeneous classes supported, including pedestrians and cyclists	Homogeneous vehicles other than size and behavioural parameters
Roads	Complex roads including sidewalks, bike lanes, ramps, roundabouts	Simpler roads with lanes being homogeneous other than width and speed
Traffic signals	Detailed signal plans including yellow change and pedestrian intervals	Binary green/red for roadlinks to determine status of traffic lights
Randomness	Incorporates random perception error into vehicle behaviour	Largely deterministic other than vehicle priority determination

Table 4.1: Comparison of implementations of SUMO and CityFlow.

4.4 Varying Driver Behaviour

Addressing Research Question 4.2 requires distributional equivalence to be assessed under different variations of driver behaviour; as established in Section 4.2.2, these are implemented through car-following and lane-changing models in traffic simulators. In both CityFlow and SUMO, car-following logic is used to maintain a safe gap to the leading vehicle, while lane changes are used to switch vehicles between lanes so that they can take appropriate roadlinks at intersections to continue their routes. However, in CityFlow, car-following and lane-changing are handled in separate threads; in SUMO, they are handled in sequence.

4.4.1 Car-Following Models

Car-following models determine the speeds at which vehicles travel unobstructed (free speed), follow a lead vehicle (following speed), and stop at an obstacle (stopping speed). Two types of numerical integration can be used to calculate vehicle speeds: a Euler update, which solves for the speed at discrete timesteps, and a ballistic update, which solves for the acceleration at discrete timesteps and applies it to the speed. I controlled both simulators to use ballistic updates.

Several shared parameters have varying effects on different car-following models. CityFlow assumes that vehicles have usual and maximum accelerations and decelerations; SUMO assumes that vehicles have a maximum possible *acceleration* and *deceleration* (which are used as the usual values), and a maximum *emergency deceleration*. I used the latter formulation here, although this may have resulted in more aggressive behaviour than if lower usual accelerations and decelerations were used. Additionally, both simulators model vehicles as having minimum desired following distances in terms of space (i.e. the *minimum gap*) and time (i.e. the *minimum headway*). Capela Dias et al. [48] co-varied these parameters to model the effect of driver aggressiveness on travel time. I adopted their taxonomy of aggressiveness types, but excluded gender effects due to disagreement in the literature on their significance [364, 367]. This gave six parameter settings (Table 4.2).

Table 4.2: Parameter settings for six aggressiveness types based on Capela Dias et al. [48]. I set maximum emergency deceleration to $-9.0\,\mathrm{m/s^2}$, as they did not specify this parameter.

Type	Max. accel.	Max. decel.	Max. emerg. decel.	Min. gap	Min. headway
Aggressive young	$3.1\,\mathrm{m/s^2}$	$-5.5\,\mathrm{m/s^2}$	$-9.0\,\mathrm{m/s^2}$	1.2 m	1.0 s
Courteous young	$2.5\,\mathrm{m/s^2}$	$-4.5\,\mathrm{m/s^2}$	$-9.0\,\mathrm{m/s^2}$	2.5 m	1.0 s
Aggressive middle-aged	$2.9\,\mathrm{m/s^2}$	$-5.0\,\mathrm{m/s^2}$	$-9.0\,\mathrm{m/s^2}$	2.0 m	1.3 s
Courteous middle-aged	$2.4\,\mathrm{m/s^2}$	$-4.1\,\mathrm{m/s^2}$	$-9.0\,\mathrm{m/s^2}$	2.5 m	1.5 s
Aggressive old	$2.6\,\mathrm{m/s^2}$	$-4.5\,\mathrm{m/s^2}$	$-9.0\,\mathrm{m/s^2}$	2.0 m	1.7 s
Courteous old	$2.3\,\mathrm{m/s^2}$	$-3.8\,\mathrm{m/s^2}$	$-9.0\,\mathrm{m/s^2}$	2.5 m	1.9 s

The default car-following models in CityFlow and SUMO are both modified from the collision avoidance model of Krauß [186]. For free speed, CityFlow's implementation uses the maximum speed, while SUMO's implementation modulates this by the visible lookahead distance. For stopping speed, both simulators solve somewhat different quadratic equations to determine the deceleration needed to stop within a fixed distance. For following speed, a target distance is maintained, which is computed by the desired minimum headway as well as the speed and maximum deceleration of the lead vehicle. I added SUMO's variant of the Krauß model to CityFlow. SUMO implements several other car-following models, of which I also re-implemented several in CityFlow to introduce variation in driver behaviour:

- The collision avoidance/action point model of Wagner [384], which probabilistically combines Krauß's model with the action point model of Todosiev [370].
- The collision avoidance model of Wiedemann [407], a behavioural model that varies between free, approaching, following, and emergency modes based on the gap to the lead vehicle.
- The adaptive cruise control (ACC) model of Milanés and Shladover [251], which determines acceleration using a "speed control" method if the gap to the lead vehicle is large and using a "gap control" method if the gap is small.

4.4.2 Lane-Changing Models

Lane-changing models are rule-based in CityFlow and SUMO. In both CityFlow and SUMO, a vehicle initiates a lane change by signalling the lead and lag vehicles on the destination lane. If the gaps to the lead and lag vehicles are sufficiently large (Figure 4.3), the vehicle makes the lane change. Upon completion of the lane change, the vehicle is instantly teleported. (SUMO also supports a sublane model [19, 328] that models lateral movement, which I excluded from consideration as a control in my experiments.) Despite these similarities, the two simulators'

lane-changing models differ irreconcilably in implementation details. Vehicles in CityFlow only perform lane changes to follow their routes, and lane-changing is based on the explicit insertion of a copy of the vehicle (the "shadow vehicle") on the target lane that the lag vehicle will follow. Vehicles in SUMO follow a parametrised hierarchy of motivations, which includes strategic lane changes to follow routes as well as tactical lane changes for overtaking [98].

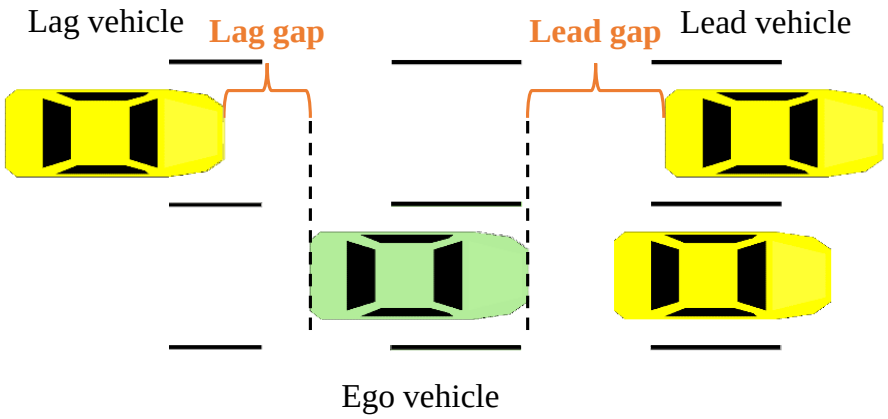


Figure 4.3: Diagram showing lead and lag gaps in lane-changing.

I introduced one of SUMO's parameters, the gap tolerance factor, to CityFlow. When vehicles are changing lanes, the minimum gap required to initiate a lane change is given by the necessary gap for collision avoidance divided by this constant factor. A factor of 1.0 represents the default behaviour. My experiments varied this factor between 0.5, 0.82, 1.0, 1.18, and 1.5, representing decreases/increases of 18% and 50% in tolerance. This is based on Sun and Kondyli [354], who observed the mean gaps for forced, cooperative, and free lane change manoeuvres to be 45 ft, 53 ft, and 109 ft in a video dataset.

4.5 Experimental Setup

To address the research questions, I designed two experiments, the basic structure of which is summarised in Figure 4.4. Both address Research Question 4.1 and Research Question 4.2 by assessing CityFlow and SUMO's distributional equivalence under different settings of driver behaviour. The two experiments also address separate aspects of Research Question 4.3: Experiment 1 assesses distributional equivalence while varying the simulation scale by traffic demand, and Experiment 2 while varying it by road network size. I accomplished this using road networks from the benchmark dataset of Ault and Sharon [14]. All of these networks were initially defined using SUMO syntax; I used the SUMO-to-CityFlow network converter of Zhang et al. [442] to generate their CityFlow counterparts, and created a flow converter to map between flows in the two simulators.

I considered four independent variables in both experiments: the car-following model (Section 4.4.1, 5 levels), the car-following aggressiveness parameters (Section 4.4.1, 6 levels), the

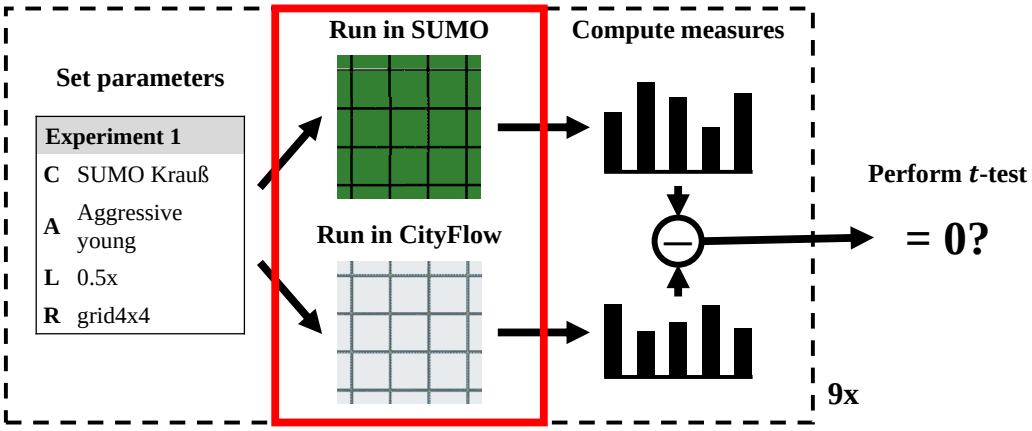


Figure 4.4: Diagram of experimental flow.

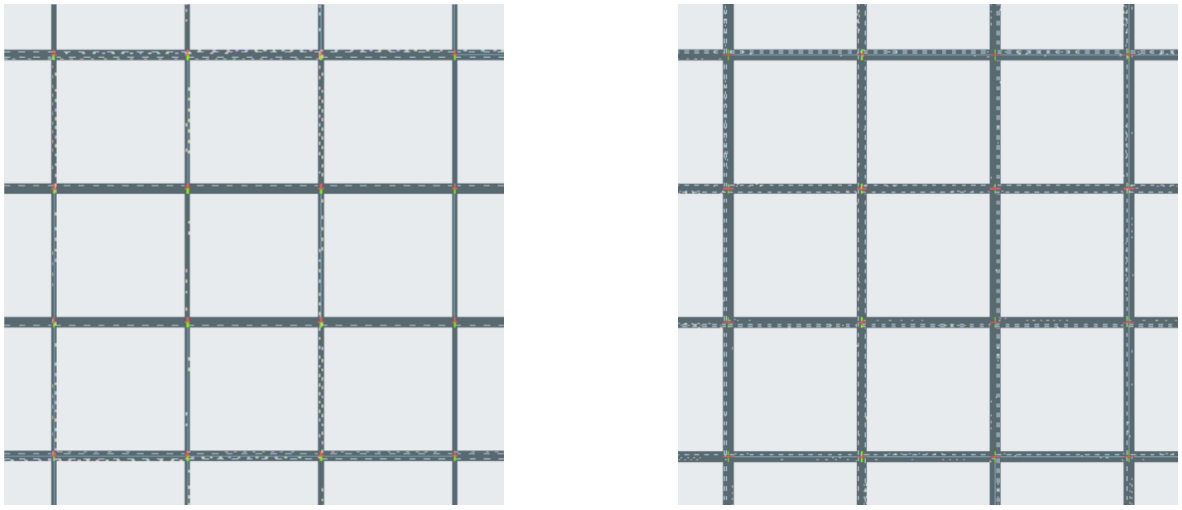
lane-changing gap tolerance (Section 4.4.2, 5 levels), and the road network (2 levels). For the car-following model, I compared the default and SUMO-based implementations of the Krauß model in CityFlow to the Krauß model in SUMO, while I compared the other models' CityFlow implementations to their respective SUMO counterparts. I also held constant the fundamental lane-changing models for the two simulators, as was the traffic signal program — a simple fixed-time program retained from the original dataset.

I used eight instantiations of two types of measures to assess distributional equivalence. First, the root mean squared error (RMSE) quantifies the point-to-point difference in individual outcome measures. I computed this as the mean RMSE of the total travel time and waiting time (defined as time that a vehicle spends queued with a speed $< 0.1^{\mathrm{m}}/_{\mathrm{s}}$) over all vehicles and timesteps; the mean RMSE of per-lane total vehicle counts and queued vehicle counts over all lanes and timesteps; and the mean RMSE of the speed and acceleration over all individual vehicles and timesteps. Second, the Kullback-Liebler (KL) divergence measures the difference in the distribution of outcomes over the entire road network. I computed this for the distributions of vehicle counts and queued vehicle counts as the mean over all timesteps.

4.5.1 Experiment 1: Traffic Demand

For this experiment, I used the road networks arterial4x4 [226] and grid4x4 [54] (Figure 4.5) from Ault and Sharon [14]'s RESCO benchmark. Both are synthetic grid networks with similar topologies, but they vary in the level of congestion. grid4x4 has six-lane roads with a uniformly distributed demand of 1,473 vehicles. arterial4x4 has major (four-lane) and minor (two-lane) roads, and a demand of 2,484 vehicles that alternates between major and minor roads. arterial4x4's traffic pattern leads to congestion and degraded RL performance in simulations [14].

The following power analysis uses these variable names: C for car-following models, A for car-following aggressiveness, L for lane-changing gap tolerance, and R for road network. The second-order linear multiple regression included 5 (C) + 6 (A) + 1 (L) + 1 (L^2) + 2 (R) + 10



(a) arterial4x4 (b) grid4x4

Figure 4.5: Screenshots in CityFlow of the arterial4x4 and grid4x4 road networks.

$(C \cdot R)$ + 12 $(A \cdot R)$ + 2 $(L \cdot R)$ + 30 $(C \cdot A)$ + 5 $(C \cdot L)$ + 6 $(A \cdot L)$ = 80 variables. Using G*Power 3.1.9.7's power calculation for ordinary linear multiple regression with a fixed model and R^2 increase, a small effect size of 0.02, and $\alpha = \beta = 0.95$, I computed the total necessary sample size as 2,646. Divided by the number of cells, $5 \cdot 6 \cdot 5 \cdot 2 = 300$, I computed the number of replications per cell as $\lceil \frac{2{,}646}{300} \rceil = 9$. I executed all replications with Python 3.9.16, SUMO 1.12.0, and a modified version of CityFlow 0.1 on a shared server with four cores, two 4.2GHz Intel i7-7700K processors per core, and 62 GiB of RAM.

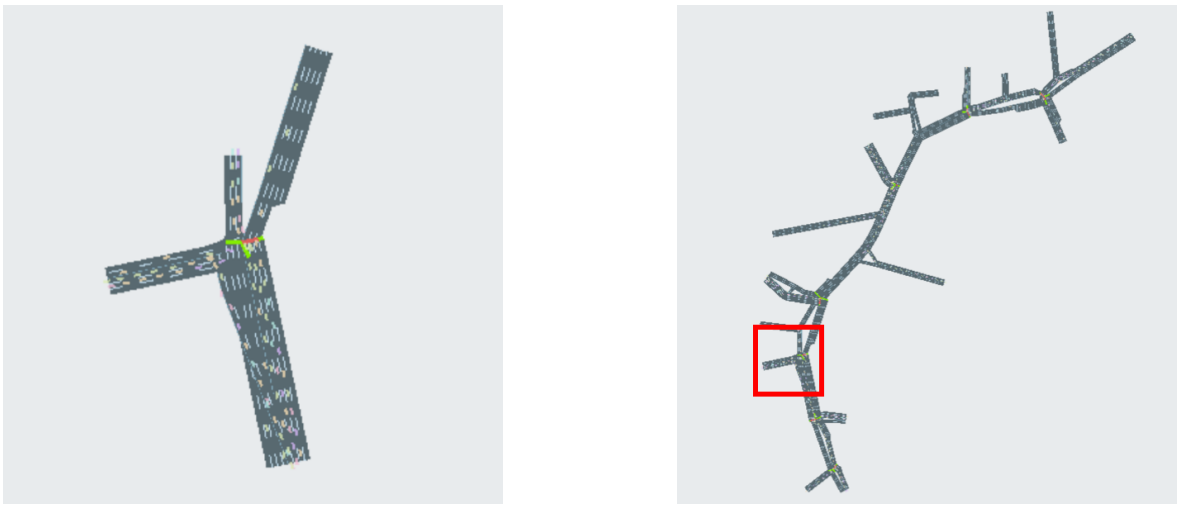
4.5.2 Experiment 2: Network Scale

For this experiment, I used the road networks ingolstadt1 and ingolstadt7 (Figure 4.6) from Ault and Sharon [14]'s RESCO benchmark. Both are subsets of the Ingolstadt road network that was simulated by Lobo et al. [218]. They respectively contain 1 and 7 signalised intersections, representing a single busy intersection and a larger arterial road; ingolstadt7 is a superset of ingolstadt1. The total demands of the two road networks are respectively 1,716 and 3,031 vehicles.

Based on the analysis in Section 4.5.1, I computed the number of replications per cell as $\lceil \frac{2{,}646}{300} \rceil = 9$. I executed all replications using Python 3.9.16, SUMO 1.12.0, and a modified version of CityFlow 0.1 on a shared server with four cores, two 4.2GHz Intel i7-7700K processors per core, and 62 GiB of RAM.

4.6 Experimental Results

One-sample t-tests indicated that all of the RMSE and KL divergence measures were significantly different from 0, with a p-value $\ll 0.001$ for all cells in Experiments 1 and 2. This suggests a lack of distributional equivalence between CityFlow and SUMO. The following subsections explore the



(a) ingolstadt1 (b) ingolstadt7

Figure 4.6: Screenshots in CityFlow of the ingolstadt1 and ingolstadt7 road networks.

results of second-order linear multiple regression for each of the measures. Notably, coefficients involving the difference between the SUMO and CityFlow Krauß model implementations were generally not significant.

In the following tables, abbreviations of variables follow Section 4.5. Subscripts denote the values of the levels. Car-following models are abbreviated as "dK" = default Krauß, "SK" = SUMO Krauß, "Wa" = Wagner, "Wi" = Wiedemann. Aggressiveness types are abbreviated as in Table 4.2. Road networks are abbreviated as "G" = grid4x4, "I7" = ingolstadt7.

4.6.1 Experiment 1: Traffic Demand

For total time and waiting time RMSEs, the road network, car-following model, and aggressiveness generally had significant effects, as did various pairwise interactions between them. The uncongested grid4x4 network had significantly lower RMSEs (coefficients: $-890.61/-1035.46$) than the congested arterial4x4 network (intercepts: $1718.61/1815.46$), suggesting that congestion worsened the distributional equivalence of these measures. In arterial4x4, the Wagner (coefficients relative to SUMO Krauß: $546.8/466.65$) and Wiedemann models (coefficients: $100.96/208.99$) had significantly higher RMSEs; these differences were smaller for grid4x4 (coefficients relative to SUMO Krauß: $49.5/33.75$ for Wagner; $48.91/34.85$ for Wiedemann).

For total and queued vehicle count RMSEs and KL divergences, I generally found significant effects for the road network, car-following model, aggressiveness, and gap tolerance, along with various pairwise interactions between them. The RMSEs and KL divergences showed distinct patterns: the RMSEs were much lower for grid4x4 than arterial4x4 (coefficients: $-5.15/-4.93$), but the KL divergences had less variation (coefficients: $-0.621/-0.013$). Yet, the KL divergences had low enough standard deviations that the road network's effects remained significant. High aggressiveness in arterial4x4 generally yielded higher RMSEs (coefficients for aggressive young relative

to aggressive middle-aged: 2.16/2.46) and KL divergences (coefficients: 0.228/0.139). While the same was true for the RMSEs in grid4x4, its KL divergences were lower for more aggressive settings (coefficients: $-0.204/-0.122$). Despite higher time measures, the Wagner model led to lower measures for total vehicle count (coefficients relative to SUMO Krauß: $-1.16/-0.193$).

For vehicle speed and acceleration RMSEs, the road network, car-following model, and aggressiveness generally had significant effects, as did various pairwise interactions between them. Greater equivalence in vehicle distributions did not always correspond to more similar vehicle-level measures. Both speed and acceleration RMSEs increased for grid4x4 (coefficients: 2.62/0.418) even though the other measures were lower on average. Also, unlike its vehicle count measures but like its time measures, the Wagner model had significantly higher speed and acceleration RMSEs (coefficients relative to SUMO Krauß: 2.71/1.26).

4.6.2 Experiment 2: Network Scale

For total time and waiting time RMSEs, the aggressiveness and its interactions generally had significant effects, along with the Wiedemann model and its interactions. The best-fitting model for total time did not include a road network-gap tolerance interaction, whereas the model for waiting time did. The smaller ingolstadt1 network had lower but more variable RMSEs (intercepts: 899.29/1170.3), while the larger ingolstadt7 network had significantly higher but more uniform RMSEs (coefficients: 1264.59/1046.07). For ingolstadt1, the most aggressive parameter settings led to significantly higher RMSEs (coefficients of aggressive young relative to aggressive middle-aged: 408.36/568.65). Likewise, the Wiedemann model had significantly higher RMSEs than other car-following models in ingolstadt1 (coefficients relative to SUMO Krauß: 192.29/1917.31), but this effect was reversed for ingolstadt7 (coefficients: $-349.13/-464.13$).

For total and queued vehicle count RMSEs and KL divergences, the road network and aggressiveness generally had significant effects, as did various pairwise interactions between them and with the gap tolerance. Unlike Experiment 1, the RMSE measures were lower in ingolstadt7 than in ingolstadt1 (coefficients: $-2/-1.67$), but the KL divergence measures were higher (coefficients: 0.461/2.48). More aggressive parameter settings again led to significant increases in the RMSEs and KL divergences, with a larger increase in RMSEs (coefficients of aggressive young relative to aggressive middle-aged: 1.35/2.29) than in KL divergences (coefficients: 0.595/0.717). However, the increase in both measures was smaller in ingolstadt7 (coefficients: 0.764/1.507 for RMSE, 0.283/0.413 for KL divergences). Both the Wagner and ACC models had RMSEs that significantly increased with gap tolerance (coefficients relative to SUMO Krauß per unit of gap tolerance: 0.151/0.056 for Wagner, 0.165/0.092 for ACC), but KL divergences that significantly decreased with it (coefficients: $-0.285/-0.511$ for Wagner, $-0.354/-0.642$ for ACC).

For vehicle speed and acceleration RMSEs, the Wagner and Wiedemann models along with the aggressiveness had significant effects, as did various pairwise interactions of the road network with the car-following models and aggressiveness. Again, the road network had significant effects on both speed and acceleration RMSEs, with these measures being higher for ingolstadt7 (coefficients: 5.51/0.173). The Wagner car-following model had significantly higher RMSEs (coefficients relative to SUMO Krauß: 0.363/0.461), whereas the Wiedemann model had significantly lower RMSEs (coefficients: $-1.94/-0.957$).

Table 4.3: Fitted regression coefficients for Experiment 1. Default levels are C = ACC, A = AM, R = arterial4x4. N/A denotes variables not included in the response surface. Significance levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

Dep. var.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Intercept	970.12***	923.27***	4.05***	2.58***	3.27***	2.43***	2.82***	4.92***
R_{G}	-940.8***	-928.6***	-3.4***	-0.289***	-2.96***	0.098***	2.1***	0.021
C_{dK}	-189.49***	-260.91***	2.61***	0.098***	1.94***	0.028*	0.821***	0.1***
C_{SK}	-192.08***	-263.95***	2.6***	0.102***	1.93***	0.029*	0.822***	0.119***
C_{Wa}	-788.82***	-734.96***	-2.03***	-0.291***	-2.05***	-0.224***	2.11***	1.48***
C_{Wi}	523.87***	528.37***	0.964***	0.446***	1.21***	0.394***	-0.309***	-0.073***
A_{AO}	15.1	-9.09	-0.026	0.082***	-0.182***	0.084***	-0.265***	-0.136***
A_{AY}	114.09***	121.5***	0.512***	0.002	0.631***	0.006	0.113***	0.067**
A_{CM}	-26.02	-62.04*	-0.143***	0.024**	-0.211***	0.038***	-0.291***	-0.175***
A_{CO}	-34.31	-121.33***	-0.154***	0.056***	-0.463***	0.074***	-0.408***	-0.194***
A_{CY}	-3.25	-7.94	-0.076*	-0.018*	0.056	-0.013	-0.135***	-0.046*
$R_{\mathrm{G}}C_{\mathrm{dK}}$	154.88***	229.28***	-2.61***	-0.081***	-1.94***	-0.012	-0.766***	-0.125***
$R_{\mathrm{G}}C_{\mathrm{SK}}$	155.5***	230.53***	-2.61***	-0.084***	-1.94***	-0.01	-0.764***	-0.125***
$R_{\mathrm{G}}C_{\mathrm{Wa}}$	775.41***	702.88***	1.81***	0.241***	1.79***	0.167***	-2.15***	-1.45***
$R_{\mathrm{G}}C_{\mathrm{Wi}}$	-267.66***	-144.34***	-0.775***	-0.315***	-0.564***	-0.265***	0.344***	0.008
$R_{\mathrm{G}}A_{\mathrm{AO}}$	-48.38***	-17.35	-0.087**	-0.087***	0.088*	-0.038***	0.179***	0.072***
$R_{\mathrm{G}}A_{\mathrm{AY}}$	-60.32***	-75.11***	-0.43***	-0.004	-0.521***	0.069***	-0.234***	-0.15***
$R_{\mathrm{G}}A_{\mathrm{CM}}$	-27.02*	-2.7	0.088**	-0.031***	0.162***	0.036***	0.278***	0.11***
$R_{\mathrm{G}}A_{\mathrm{CO}}$	-61.15***	17.82	-0.012	-0.058***	0.26***	0.001	0.418***	0.157***
$R_{\mathrm{G}}A_{\mathrm{CY}}$	-26.89*	-45.34**	0.134***	0.045***	-0.05	0.075***	0.136***	0.021
$C_{\mathrm{dK}}A_{\mathrm{AO}}$	45.19**	10.91	-0.038	-0.007	-0.159*	-0.028*	-0.096***	0.067***
$C_{\mathrm{SK}}A_{\mathrm{AO}}$	47.08**	12.14	-0.034	-0.012	-0.158*	-0.032**	-0.13***	0.058**
$C_{\mathrm{Wa}}A_{\mathrm{AO}}$	-4.14	10.12	0.542***	0.069***	0.601***	0.095***	0.147***	-0.007
$C_{\mathrm{Wi}}A_{\mathrm{AO}}$	40.85*	47.8*	0.103*	-0.006	0.194**	-0.023*	0.056*	0.086***
$C_{\mathrm{dK}}A_{\mathrm{AY}}$	-64.3***	-52.04*	0.058	-0.033**	0.072	-0.001	0.192***	-0.003
$C_{\mathrm{SK}}A_{\mathrm{AY}}$	-61.46***	-48.95*	0.064	-0.035***	0.076	-0.004	0.178***	-0.011
$C_{\mathrm{Wa}}A_{\mathrm{AY}}$	-71.42***	-74.05**	-0.486***	-0.007	-0.556***	-0.005	-0.165***	-0.034*
$C_{\mathrm{Wi}}A_{\mathrm{AY}}$	-56.73**	-50.16*	-0.087	-0.004	-0.163*	0.015	-0.001	0.003
$C_{\mathrm{dK}}A_{\mathrm{CM}}$	30.41	28.05	-0.087	-0.019*	-0.166**	-0.041***	-0.104***	0.073***
$C_{\mathrm{SK}}A_{\mathrm{CM}}$	34.88*	32.75	-0.08	-0.023*	-0.16*	-0.045***	-0.113***	0.066***
$C_{\mathrm{Wa}}A_{\mathrm{CM}}$	13.33	42.11	0.479***	0.127***	0.533***	0.158***	0.168***	0.031
$C_{\mathrm{Wi}}A_{\mathrm{CM}}$	80.26***	96.42***	0.005	0.032**	0.072	-0.007	0.001	0.118***
$C_{\mathrm{dK}}A_{\mathrm{CO}}$	116.82***	90.82***	-0.074	-0.005	-0.153*	-0.021	-0.193***	0.074***
$C_{\mathrm{SK}}A_{\mathrm{CO}}$	115.25***	89.26***	-0.074	-0.008	-0.155*	-0.022	-0.2***	0.072***
$C_{\mathrm{Wa}}A_{\mathrm{CO}}$	68.58***	124.53***	0.991***	0.133***	1.17***	0.181***	0.237***	-0.015
$C_{\mathrm{Wi}}A_{\mathrm{CO}}$	81.46***	98.69***	0.039	0.005	0.195**	-0.025*	0.041	0.134***
$C_{\mathrm{dK}}A_{\mathrm{CY}}$	-0.113	19.28	-0.041	-0.044***	0.098	-0.025*	-0.018	0.004
$C_{\mathrm{SK}}A_{\mathrm{CY}}$	3.37	22.38	-0.031	-0.044***	0.108	-0.023*	-0.022	0.006
$C_{\mathrm{Wa}}A_{\mathrm{CY}}$	-3.99	0.919	-0.089*	0.003	-0.128*	0.017	-0.013	0.033
$C_{\mathrm{Wi}}A_{\mathrm{CY}}$	39.96*	54.56*	-0.081	0.026**	-0.068	0.024*	-0.029	0.068***
L	-87.49***	-120.24***	-0.038	-0.019**	-0.167***	-0.022**	-0.062**	-0.034*
$R_{\mathrm{G}}L$	123.75***	180.39***	0.068**	0.041***	0.286***	0.05***	0.041**	0.026**
$C_{\mathrm{dK}}L$	13.24	15.42	0.028	0.003	0.049	0.001	-0.021	-0.014
$C_{\mathrm{SK}}L$	13.38	15.46	0.034	0.001	0.054	0.001	-0.009	-0.025
$C_{\mathrm{Wa}}L$	12.96	14.77	-0.021	-0.006	-0.015	-0.019	-0.028	-0.034*
$C_{\mathrm{Wi}}L$	-287.21***	-425.27***	-0.187***	-0.142***	-0.68***	-0.129***	-0.047*	-0.004
$A_{\mathrm{AO}}L$	8.21	9.99	N/A	N/A	N/A	N/A	0.08**	0.013
$A_{\mathrm{AY}}L$	-5.87	-3.05	N/A	N/A	N/A	N/A	0.056*	0.068***
$A_{\mathrm{CM}}L$	25.87	26.58	N/A	N/A	N/A	N/A	0.048*	0.033*
$A_{\mathrm{CO}}L$	24.55	24.98	N/A	N/A	N/A	N/A	0.05*	0.021
$A_{\mathrm{CY}}L$	26.82	35.96	N/A	N/A	N/A	N/A	0.047	0.034*

Table 4.4: Fitted regression coefficients for Experiment 2. Default levels are C = ACC, A = AM, R = ingolstadt1. N/A denotes variables not included in the response surface. Significance levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

Dep. var.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Intercept	7.6	22.81	2.28***	3.07***	1.21***	2.68***	5.8***	1.75***
R_{I7}	426.11***	367.35***	-0.545***	1.71***	0.091*	2.44***	-0.476***	-0.025
C_{dK}	61.93	68.9	0.924***	-0.018	1.05***	0.099*	-0.202*	-0.033
C_{SK}	47.15	51.86	0.87***	-0.021	0.993***	0.104*	-0.126	-0.023
C_{Wa}	-15.79	1.18	-0.637***	-0.616***	-0.19**	-0.378***	0.545***	0.473***
C_{Wi}	1389.63***	1917.31***	2.24***	0.035	3.38***	-0.169***	0.246**	-0.302***
A_{AO}	28.69	31.65	0.317***	0.058	0.222**	0.031	0.035	-0.129***
A_{AY}	-84.5*	-90.65*	-0.384***	-0.104**	-0.419***	-0.194***	0.315***	0.157***
A_{CM}	12.43	16.06	0.366***	-0.177***	0.383***	-0.17***	-0.719***	-0.321***
A_{CO}	34.07	33.49	0.44***	-0.085*	0.36***	-0.073	-0.579***	-0.339***
A_{CY}	-47.2	-48.54	-0.104	-0.014	-0.054	0.135**	-0.553***	-0.171***
$R_{\text{I7}}C_{\text{dK}}$	-92.99***	-79.78***	-0.913***	-0.209***	-0.889***	-0.291***	0.28***	0.075***
$R_{\text{I7}}C_{\text{SK}}$	-113.04***	-102.86***	-0.936***	-0.219***	-0.917***	-0.295***	0.284***	0.087***
$R_{\text{I7}}C_{\text{Wa}}$	-104.46***	-99.8***	0.051	0.423***	-0.257***	0.248***	1.01***	0.092***
$R_{\text{I7}}C_{\text{Wi}}$	171.07***	-49.3*	-0.812***	0.156***	-1.44***	-0.11***	0.564***	-0.071***
$R_{\text{I7}}A_{\text{AO}}$	-39.49	-54.69*	-0.3***	-0.07***	-0.221***	-0.114***	0.055	0.04***
$R_{\text{I7}}A_{\text{AY}}$	-143.27***	-182.13***	-0.066*	0.092***	-0.164***	0.141***	-0.099*	0.093***
$R_{\text{I7}}A_{\text{CM}}$	214.11***	267.24***	-0.04	0.059**	0.093*	0.056*	-0.343***	-0.162***
$R_{\text{I7}}A_{\text{CO}}$	153.16***	201.83***	-0.224***	0.042*	-0.032	0.033	-0.489***	-0.144***
$R_{\text{I7}}A_{\text{CY}}$	210.73***	260.83***	0.151***	0.047**	0.203***	-0.033	-0.228***	-0.157***
$C_{\text{dK}}A_{\text{AO}}$	-24.91	-29.01	0.125*	0.03	0.037	-0.021	-0.023	-0.044*
$C_{\text{SK}}A_{\text{AO}}$	1.49	0.955	0.157**	0.045	0.074	-0.015	0.017	-0.041*
$C_{\text{Wa}}A_{\text{AO}}$	-40.91	-29.49	-0.141**	-0.048	-0.103	0.068	-0.219**	-0.056**
$C_{\text{Wi}}A_{\text{AO}}$	11.65	39.36	-0.083	-0.056*	-0.036	-0.051	0.253***	0.024
$C_{\text{dK}}A_{\text{AY}}$	62.86	68.72	-0.086	-0.057*	0.013	-0.064	-0.09	0.037*
$C_{\text{SK}}A_{\text{AY}}$	92.72**	107.09**	-0.023	-0.04	0.082	-0.044	-0.06	0.033
$C_{\text{Wa}}A_{\text{AY}}$	46.29	52.08	0.072	-0.064*	0.105	-0.155***	-0.208**	-0.023
$C_{\text{Wi}}A_{\text{AY}}$	148.65***	213.29***	0.608***	0.052	0.712***	-0.081*	0.085	-0.052**
$C_{\text{dK}}A_{\text{CM}}$	68.22*	69.16	0.359***	0.113***	0.309***	0.054	0.469***	0.039*
$C_{\text{SK}}A_{\text{CM}}$	87.22**	92.12*	0.385***	0.135***	0.334***	0.074*	0.545***	0.048**
$C_{\text{Wa}}A_{\text{CM}}$	79.84*	78.72*	0.049	0.231***	0.04	0.297***	0.618***	0.074***
$C_{\text{Wi}}A_{\text{CM}}$	-164***	-211.64***	-0.402***	-0.072**	-0.483***	0.047	-0.238**	0.035*
$C_{\text{dK}}A_{\text{CO}}$	21.83	19.98	0.407***	0.07*	0.297***	-0.003	0.448***	0.015
$C_{\text{SK}}A_{\text{CO}}$	34.23	35.11	0.424***	0.068*	0.322***	-0.017	0.45***	0.013
$C_{\text{Wa}}A_{\text{CO}}$	43.95	47.56	-0.042	0.156***	-0.02	0.234***	0.457***	0.021
$C_{\text{Wi}}A_{\text{CO}}$	-164.22***	-204.38***	-0.464***	-0.105***	-0.508***	0.04	-0.257***	0.023
$C_{\text{dK}}A_{\text{CY}}$	0.075	-4.09	-0.055	0.055*	-0.029	0.037	0.4***	0.104***
$C_{\text{SK}}A_{\text{CY}}$	21.65	21.13	-0.043	0.037	-0.011	0.012	0.454***	0.115***
$C_{\text{Wa}}A_{\text{CY}}$	53	44.29	0.034	0.114***	0.009	0.012	0.57***	0.056**
$C_{\text{Wi}}A_{\text{CY}}$	-188.69***	-256.56***	-0.374***	-0.128***	-0.476***	-0.048	-0.495***	-0.02
L	52.1	32.01	-0.081	0.011	-0.105*	-0.031	-0.031	0.011
$R_{\text{I7}}L$	-105.17***	-48.53*	0.092**	-0.02	0.155***	0.022	N/A	0.039***
$C_{\text{dK}}L$	-49.19	-58.58	-0.429***	0.06*	-0.459***	0.14***	0.047	0.027
$C_{\text{SK}}L$	-53.48	-64.42*	-0.414***	0.051*	-0.445***	0.121***	-0.048	0.014
$C_{\text{Wa}}L$	-25.45	-33.98	0.007	-0.091***	0.002	-0.07*	-0.103	-0.023
$C_{\text{Wi}}L$	-694.07***	-923.7***	-0.568***	-0.042	-0.817***	0.17***	-0.059	0.2***
$A_{\text{AO}}L$	-13.86	-27.5	-0.084	-0.017	-0.105	-0.032	-0.001	0.002
$A_{\text{AY}}L$	34.56	39.67	0.166**	0.012	0.17**	0.01	-0.022	-0.036*
$A_{\text{CM}}L$	-24.34	-32.66	-0.178***	0.119***	-0.227***	0.147***	0.302***	0.05**
$A_{\text{CO}}L$	-12.05	-21.61	-0.068	0.084**	-0.107	0.099**	0.235***	0.016
$A_{\text{CY}}L$	51.39	54.83	0.164**	0.032	0.152**	0.036	0.24***	-0.002

4.6.3 Parameter Validity

I conducted parameter validation by comparing the parameter settings for car-following and lane-changing models used in the experiments to prior literature. Overall, the dependence of these parameters on external factors such as traffic density and speed suggests that the settings used in traffic simulators should be calibrated to specific road networks and conditions. However, the settings I used remain reasonable considering the variation reported in the literature.

For maximum acceleration and deceleration, my experiments respectively varied them from $2.3\ {}^{\mathrm{m}}/_{\mathrm{s}^2}$ to $3.1\ {}^{\mathrm{m}}/_{\mathrm{s}^2}$ and from $-3.8\ {}^{\mathrm{m}}/_{\mathrm{s}^2}$ to $-5.5\ {}^{\mathrm{m}}/_{\mathrm{s}^2}$. These settings were based on Capela Dias et al. [48], with smaller values for older and less aggressive drivers (Section 4.4.1); however, they considered gender effects to be negligible. Similar values have been reported in prior work [101, 183, 269]. However, among driving simulator studies, Körber et al. [183] demonstrated an age effect opposite to that assumed by Capela Dias et al. [48]. Among real-world studies, Moon and Yi [257] found a dependence of the 95th percentile of braking decelerations on speed, and Nakagawa et al. [269] reported a significant interaction between age and gender.

For minimum gap and headway time, my experiments respectively varied them from 1.2 m to 2.5 m and from 1.0 s to 1.9 s. This again followed Capela Dias et al. [48], with larger values for older and less aggressive drivers (Section 4.4.1). Similar values have been reported in prior work [83, 247, 316]. However, among driving simulator studies, Körber et al. [183] demonstrated an age effect opposite to that assumed by Capela Dias et al. [48]. Among real-world studies, Moon and Yi [257] found a dependence of the 95th percentile of braking decelerations on speed, and Nakagawa et al. [269] reported a significant interaction between age and gender.

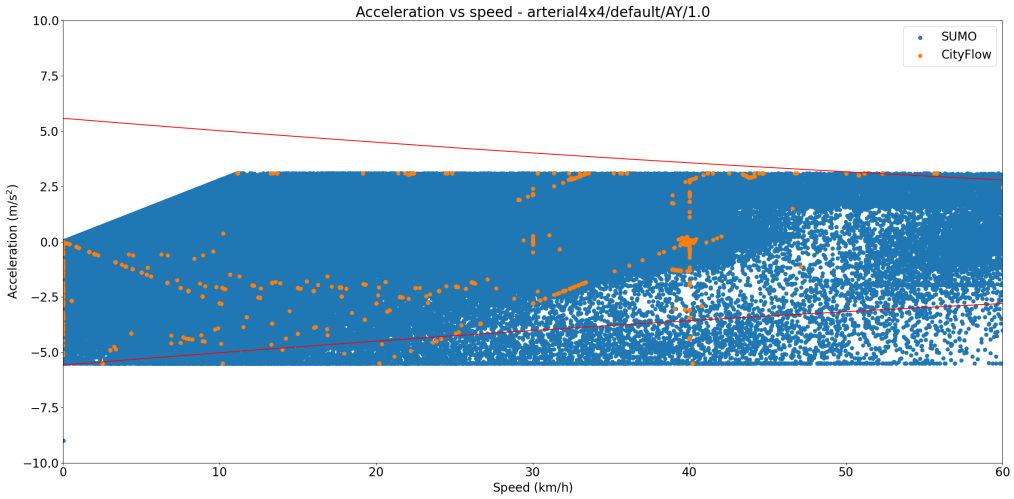
Lane-changing gap tolerance can be approximately quantified by variance in accepted gap widths from empirical lane-changing behaviour. In my experiments, I varied it upward and downward by 18% and 50%. As described in Section 4.4.2, Sun and Kondyli [354]'s real-world study reported the mean gaps for forced, cooperative, and free lane changes to be 45 ft, 53 ft, and 109 ft, representing increments of 18% and 105%. These settings are consistent with standard deviations in lead and lag gaps as reported in prior studies [17, 259]. Ali et al. [7]'s driving simulator study identified significant factors that impact gap tolerance: relative to the average male middle-aged driver, gaps are smaller for younger drivers, larger for female drivers, and smaller as speed increases. Likewise, Hill et al. [144]'s real-world study reported that the mean and standard deviation of lag gaps depended on congestion. Future work could use these factors to create a taxonomy of lane-changing behaviour similar to Capela Dias et al. [48].

4.6.4 Acceleration-Speed Diagrams

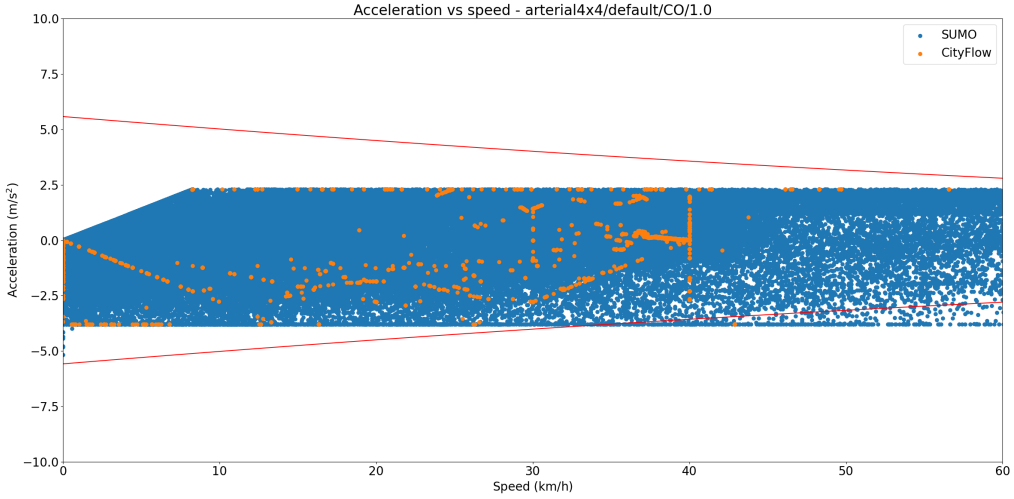
Based on movement and friction experienced by vehicles, Eboli et al. [94] derived the following upper bound on the safe magnitude of acceleration and deceleration as a function of speed:

$$|a| = g \cdot \left[0.198 \left(\frac{v}{100} \right)^2 - 0.592 \left(\frac{v}{100} \right) + 0.569 \right]$$

where a is the acceleration in $^{\mathrm{m}}/_{\mathrm{s}^2}$, v is the speed in $^{\mathrm{km}}/_{\mathrm{h}}$, and $g = 9.81\ ^{\mathrm{m}}/_{\mathrm{s}^2}$ is the gravitational acceleration. The bound yields the point at which movement and friction forces are in balance for a vehicle. I plot this bound (in red) against actual acceleration-speed points from the two simulators in Figure 4.7. The most aggressive setting of car-following parameters clearly differs from the least aggressive in that it causes vehicles to drive closer to their maximum accelerations and exceed safe deceleration limits more often at low speed. The diagrams also show that perceptual randomness in SUMO likely yielded a much wider spread of points than in CityFlow.



(a) Aggressive young (AY) drivers



(b) Courteous old (CO) drivers

Figure 4.7: Acceleration-speed diagrams comparing outcomes of CityFlow (orange) and SUMO (blue) for aggressive young (AY) and courteous old (CO) drivers under the Krauß car-following model in the arterial4x4 network.

4.7 Conclusion

In this chapter, I designed experiments to compare the low-level simulation outcomes of two traffic simulators, CityFlow and SUMO. To capture the effects of modelling real-world heterogeneity, various parameters of driver behaviour and road network scale were varied. The results indicate a lack of distributional equivalence between the simulators, with certain parameter settings worsening distributional equivalence. However, as I noted in Section 4.3, these experiments were insufficient to provide a complete characterisation of what the critical differences between these simulators are. Many aspects of CityFlow and SUMO that were not controlled — simulation control-flow, other aspects of driver behaviour, the effects of traffic signals, and randomness — could all have contributed to the observed discrepancies. Future work should perform more comprehensive, controlled evaluations of these two simulators.

Regardless, researchers in RL for TSC must not take traffic simulators for granted as a deus ex machina for training, and must recognise that they may not be interchangeable. Which simulator, then, should be chosen? I do not aim to answer this question with my work in this chapter, but some observations can be made:

- **SUMO** provides a detailed simulation that models real-world heterogeneity, and captures additional aspects of traffic management and driver behaviour
- **CityFlow** provides an efficient simulation that abstracts out and homogenises various details, reducing the number of parameters that need to be tuned

Mei et al. [243] showed that, if the `libsumo` library is used as an API, SUMO's runtime can be reduced tenfold. However, they also found that it remains slower than CityFlow for various environments. Therefore, the core trade-off between these two simulators (and others) involves realism and efficiency. There is no one best simulator; researchers must decide whether using a coarser abstraction of the environment is acceptable in exchange for faster training.

But how exactly should researchers make this decision? Crucially, RL-based signal plans may not necessarily perform better when they are trained with more granular simulators. Both introducing unnecessary complexity [456] and removing needed complexity [116] in the observation space may harm performance. Mei et al. [243] found no consistent ordering in performance across environments between RL policies trained with CityFlow or SUMO. Following their example, researchers should compare training results from different simulators and use them to design RL formulations in a principled way. One strategy may be to train a baseline using an efficient simulator (e.g. CityFlow), and then to finetune it by further training with a realistic one (e.g. SUMO).

Ultimately, the true goal is to ensure that RL-based signal plans can perform well under real-world traffic conditions, which requires that their training and validation environments are close to reality. As previously noted, distributional equivalence between simulators is a proxy measure of this goal. Unfortunately, a chicken-and-egg problem exists in that traffic simulators are intended to replace real-world data collection, yet cannot be validated without it. For now, simulations should still be developed in collaboration with stakeholders to ensure that they meet acceptable standards of fidelity. However, the future holds promise for traffic simulations that are both realistic and efficient: the increasing prevalence of connected vehicles [237] means that collection of granular real-world data for validation may be within reach.

Chapter 5

Out of the Past

An AI-Enabled Pipeline for Traffic Simulation from Noisy, Multimodal Detector Data and Stakeholder Feedback

Domain: Traffic signal control
Challenges: Uncertainty, heterogeneity

5.1 Introduction

In Chapter 4, I studied how heterogeneity is modelled by different traffic simulators, and I argued that realistic, granulator simulators are necessary to ensure that traffic signal plans created by reinforcement learning (RL) policies can perform well under real-world traffic conditions. However, a realistic *simulator* by itself is insufficient to achieve this goal, because there must also be a realistic *simulation* that is executed in the simulator to generate traffic for the policy. The construction of simulations grounded in data from physical traffic systems is an understudied problem, but it is crucial to ensuring that the environment the RL algorithm interacts with is close to reality.

Existing approaches to creating road network-scale traffic simulations have a number of limitations that hamper their realism and thus their practical applicability. Simulations based on origin-destination matrices from activity data are unrealistic and fail to make use of traffic detector data. Meanwhile, simulations based more directly on detector data have been constructed using methodologies that rely on suboptimal heuristics. All of these approaches also consider the source data to be the ground truth; they do not perform any calibration to account for sources of *uncertainty* (noise) or *heterogeneity* (multimodality) in the data. In this chapter, I address these two deployment challenges by seeking to answer the following research questions:

RESEARCH QUESTION 5.1. *Is it possible to correct for error in camera detection so that it can provide accurate vehicle counts for traffic simulation?*

RESEARCH QUESTION 5.2. *Can different sources of traffic detector data, each having different error rates and assumptions, be integrated to produce a single, unified traffic simulation?*

RESEARCH QUESTION 5.3. *How can unstructured feedback from human stakeholders be used for the calibration of traffic simulations?*

To address these research questions, I contribute a detailed pipeline (Figure 5.1) for constructing a traffic simulation from detector data. Starting from raw detector data, my pipeline consists of three steps: (1) I apply computer vision-based vehicle tracking directly to camera footage, to obtain more accurate vehicle counts than the camera detectors themselves. (2) I solve a quadratic optimization program to populate my simulation with vehicle routes. In doing so, I account for multimodality by imposing multiple sets of optimization constraints based on different sources of counts. (3) I incorporate feedback from stakeholders to refine the simulation, using a large language model (LLM)-based framework that encodes natural language into optimization constraints.

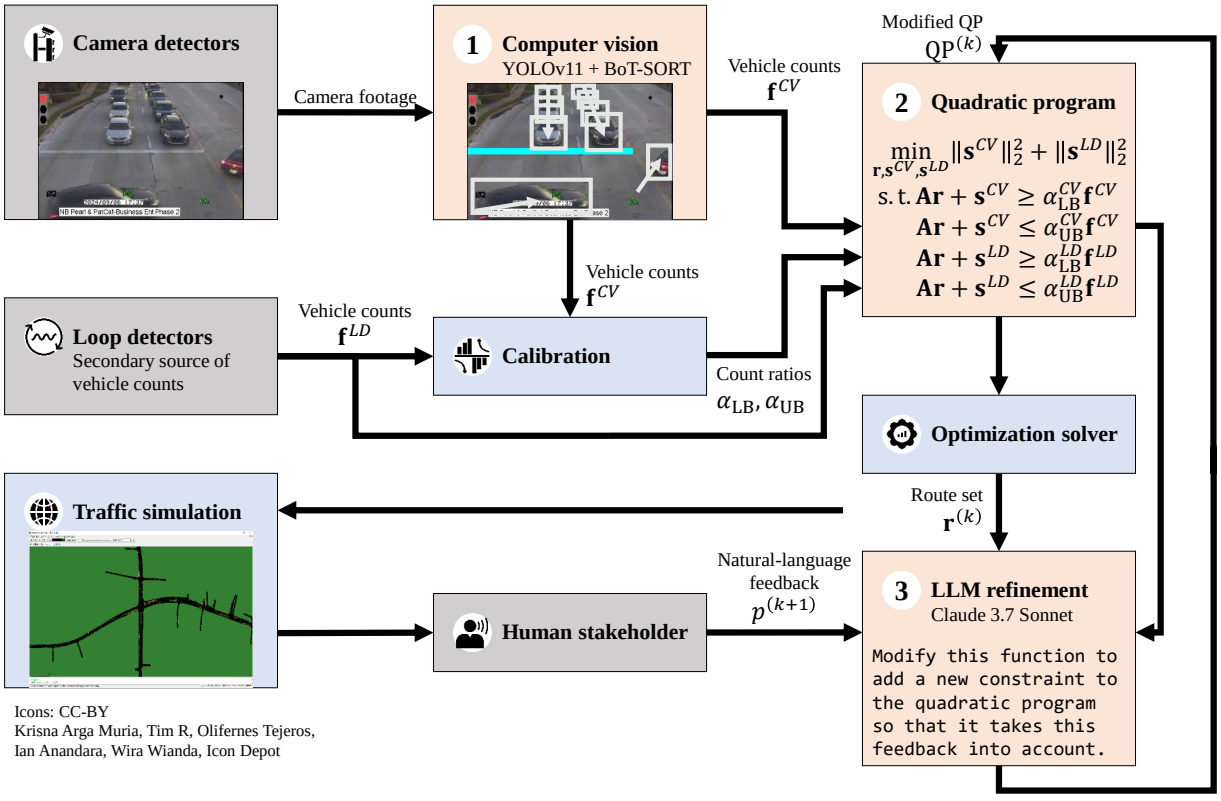


Figure 5.1: My pipeline for generating a traffic simulation from multimodal detector data.

As a proof of concept, I apply my pipeline to simulate a high-traffic road network from the city of Strongsville, Ohio. Beginning with 24 hours of recorded camera footage and detector data from 36 intersections, I created a fully realised traffic simulation. I show that: (1) My computer vision method is able to rectify systematic undercounting issues present in camera detector data. (2) My optimization method is able to generate a set of vehicle routes that is consistent with counts from both computer vision and loop detector counts, while still accounting for error in these counts. (3) My LLM framework is able to synthesise code representing sensible, quantified constraints based on qualitative stakeholder feedback.

5.2 Related Work

5.2.1 Demand Modelling for Traffic Simulation

Demand modelling, or the modelling of trips to populate a traffic simulation, is a central but difficult aspect of constructing data-driven traffic simulations [23]. One popular approach is *activity modelling*, where trips are extrapolated from censuses of the daily activities taken by a sample of households in the study region [378, 68, 69, 37, 200, 141]. While many municipalities collect this data for transportation planning, the locations of these activities are usually coarsely discretised. Furthermore, they represent a small, not necessarily representative sample of the population. This means that activity modelling-based simulations are prone to significant error [191]; thus, I do not consider activity modelling in this chapter.

An alternative approach to demand modelling directly uses data from traffic detectors. Types of detectors used for simulations include induction loops [35] and video cameras [397, 455, 417] (see further discussion in Section 2.2.2). Detectors provide accurate vehicle counts local to individual intersections, but converting them to fully-realised routes through a road network is nontrivial. Wei et al. [397], Zheng et al. [455], Xu et al. [417] all provided no details regarding how they generated vehicle routes. The route generation procedures that have been specified in prior work have relied on suboptimal heuristics. Lobo et al. [218] and Rapelli et al. [293] used detector data to adjust activity models. Bieker et al. [35] generated routes probabilistically by using turn ratios to define distributions over movements at intersections. This approach only leads to correct simulation outcomes in expectation. Finally, Qiu et al. [286]'s approach, which uses scripts included with the traffic simulator SUMO [10], is most similar to mine: they applied a two-step process of first sampling routes randomly, and then solving a linear program (LP) to approximate how many times each route should be used to match the detector counts as closely as possible. Unlike them, I solve the problem exactly as a quadratic integer program (QIP) over all possible routes.

One further limitation of previous detector-based approaches is that they rely on a single source of detector data, which is assumed to be generally error-free. Among the works cited previously, only Bieker et al. [35] reported detectors that failed to report vehicle counts and were removed from their dataset. When these works validate their simulations, they treat the detector data as the ground truth to compare their simulations against [378, 35, 68, 218, 200]. By contrast, my approach (Section 5.3.2) integrates multiple methods of processing detector data, and I adopt a semi-automated approach to validation that combines detector data with manual verification. This approach allows us to obtain higher realism in the vein of traffic simulations that rely exclusively on manual counting [263], but at a much larger scale.

5.2.2 Computer Vision for Traffic Footage

Vehicle counting can be decomposed into two distinct but related problems: *vehicle detection*, the identification of vehicles in footage; and *vehicle tracking*, the identification of these vehicles' trajectories across frames [369]. These methods can be divided into two distinct but largely disjoint waves of research. First, in the 1990s, advances in image processing led to vehicle detection methods based on extracting heuristically designed features [70]. Second, in the 2010s, the advent

of *convolutional neural networks* (CNNs) led to various deep object detection algorithms capable of automatically extracting relevant features for vehicle detection [387]. While these lines of work overlap methodologically, image processing and CNN-based methods have not been comparatively evaluated to my knowledge. I provide an in-situ evaluation of AutoScope against CNN-based vehicle counting.

5.2.3 Large Language Models for Transportation Research

Large language models (LLMs) are useful for aligning AI systems with human intuition. As such, they have been increasingly applied to the domain of transportation. One line of work has generated simulation scenarios [51, 201, 356] and reward functions for vehicular agents [136, 464] based on natural language prompts. Another line of work has used LLMs to align simulations with reality based on general knowledge; Da et al. [75] used an LLM to infer how actions taken in a simulated environment would affect a real-world environment differently.

When responding to prompts, LLMs can make use of external tools. For instance, Li et al. [201]'s agent generates and executes command-line calls to the SUMO simulator; Wang et al. [388]'s agent chooses between different perception and decision tools to perform reinforcement learning for traffic signal control. I leverage a strength of LLMs that has not been explored for transportation research to my knowledge: the synthesis of syntactically and semantically correct programs [15]. I use the LLM within an iterative framework in which the *only* external input required is qualitative natural language feedback.

5.3 Simulation Generation Pipeline

5.3.1 Computer Vision-Based Vehicle Counting from Camera Footage

In Section 2.2.2, I noted that the camera detectors deployed in most municipal transportation systems — such as those based on AutoScope — are primarily meant to detect the presence or absence of vehicles, and do not necessarily provide accurate counts. To detect individual vehicles more accurately than the camera detectors themselves, I use the YOLOv11 model [164]. In each frame of footage, the model predicts a set of bounding boxes, where each bounding box encloses an object. For each box, the model outputs a class b_c, the x and y coordinates of the centre of the bounding box (b_x and b_y), and the width and height of the box (b_w and b_h).

How can YOLO's detections of vehicles be converted to counts for each lane? I manually annotate each frame of the traffic footage with the stop bar's y-coordinate (S_y), and with x-coordinates for each lane (S_L^ℓ, S_R^ℓ). The most straightforward method to perform counting is to verify whether the bounding box for an object identified as a vehicle has crossed the position of the stop bar, i.e. I increment the count for lane ℓ when a detected vehicle has $b_x \in [S_L^\ell, S_R^\ell]$, and $|b_y - S_y| \leq \epsilon$ for some predefined threshold ϵ. When I attempted to apply this method in practice, I encountered two issues. (1) Due to instability in the real-time streaming (RTSP) connection over which I retrieve the detector footage, frames are frequently dropped. For many vehicles, this leads to the absence of the frames in which their bounding boxes' borders b_y are close to the stop bar S_y. These vehicles

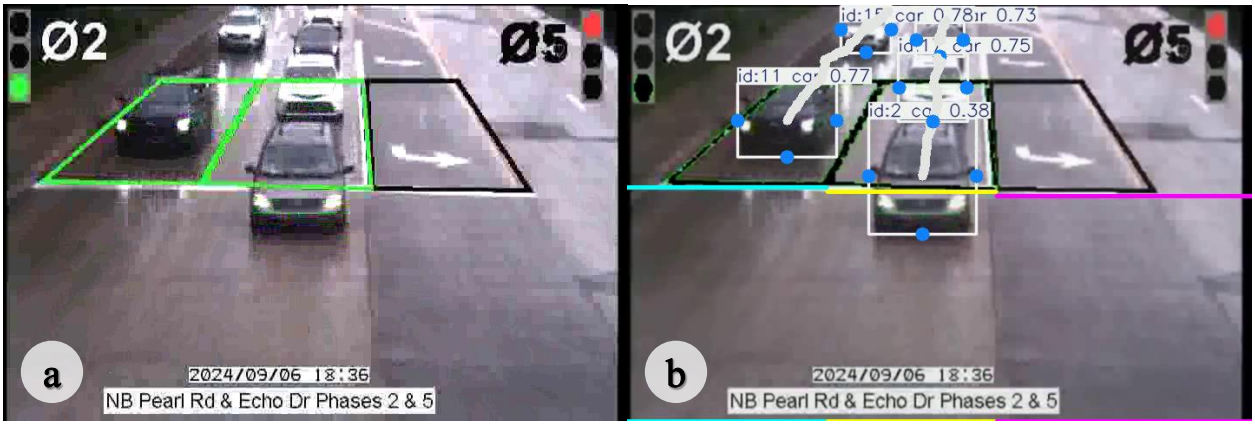


Figure 5.2: Demonstration of my computer vision vehicle counting method on camera detector footage from intersection 8 (US 42 & Echo Rd) in Strongsville, Ohio. (a) Raw footage, showing two vehicles actuating a detection zone in the centre lane. (b) Footage with preprocessing filters applied, and bounding boxes and tracks for counted vehicles annotated. The coloured lines represent manually labelled stop bar positions.

first appear far above the stop bar ($b_y \ll S_y$), and then far below the stop bar ($b_y \gg S_y$) once the footage resumes. (2) Pixelation artifacts, particularly around the detection zones marked on the footage, also obfuscate the bounding boxes.

To address dropped frames, I use the BoT-SORT algorithm [4] to perform vehicle tracking. BoT-SORT reidentifies each bounding box across consecutive frames to provide a consistent ID b_t. In each frame, for each bounding box, I verify whether b_t has already been counted. To ensure that I do not capture traffic in other directions, I only consider b_t if $b_y < S_y$ initially. If b_t has not yet been counted and $b_y > S_y$, I increment the vehicle count and mark b_t as counted. Even if the footage is missing the frame where $|b_y - S_y| \leq \epsilon$ for a vehicle, the vehicle will be counted in a later frame. I also apply filters to smooth the footage as a preprocessing step, including a non-local means filter [42], a spatiotemporal denoising filter (`hqdn3d`), a frame-blending motion interpolation filter (`minterpolate`), and a filter to remove detector actuation overlays. The results are shown in Figure 5.2b.

5.3.2 Optimization-Based Vehicle Route Generation from Multimodal Data

My computer vision method from Section 5.3.1 outputs a set of vehicle counts f_j^{CV} for a set of counting locations j. These represent the traffic volumes at the eastbound, northbound, southbound, and westbound approaches for each intersection (if they are available). I also have a set of counts from loop detectors f_j^{LD}, which overlap with the computer vision counts at a subset of counting locations. But how can these multimodal vehicle counts be integrated to generate vehicle routes for a fully-realised simulation?

I make two key assumptions to identify the set of feasible routes. (1) Given an origin and destination in the road network, I assume that vehicles perform shortest-path (Dijkstra) routing. (2) I assume that most of the traffic in the road network originates at the fringes of the network,

and few routes begin and end in the middle of a road edge (representing traffic from unmodelled driveways). This assumption holds as long as all major sources and sinks of traffic are modelled. Based on these assumptions, I enumerate the full set of routes between all counting locations, instead of randomly sampling them as in prior work.

Given this set of feasible routes, I aim to solve for the number of times each route should be used, so that the number of times they pass through the counting locations match the given vehicle counts as closely as possible. I do so by dividing 24 hours of count data into 15-minute time segments, which are indexed as $t \in \{0..95\}$. For each time segment, I match the total counts at each counting location, as well as counts for dedicated left-turn and right-turn lanes if they exist.

How closely should the counts be matched? I assume that, for some counting locations j and time segments t, there are ground truth counts f_{jt}^{M}, and that error exists in both my computer vision counts f_{jt}^{CV} and loop detector counts f_{jt}^{LD}. Let M, CV, and LD denote the sets of locations for these sources. Based on how much computer vision overcounts or undercounts for locations shared with the ground truth, I extrapolate conservative lower and upper bounds for the true counts:

$$\alpha_{\text{LB}}^{CV} = \min_{t} \min_{j \in M \cap CV} \frac{f_{jt}^{M}}{f_{jt}^{CV}}, \alpha_{\text{UB}}^{CV} = \max_{t} \max_{j \in M \cap CV} \frac{f_{jt}^{M}}{f_{jt}^{CV}}.$$

I also derive bounds for loop detectors, α_{LB}^{LD} and α_{UB}^{LD}, in a similar fashion. Then, I assume that, for time segments $t \in \{0..95\}$, $\alpha_{LB}^{CV} f_{jt}^{CV} \leq f_{jt}^{M} \leq \alpha_{UB}^{CV} f_{jt}^{CV}, \forall j \in CV$, and $\alpha_{LB}^{LD} f_{jt}^{LD} \leq f_{jt}^{M} \leq \alpha_{UB}^{LD} f_{jt}^{LD}, \forall j \in LD$.

Now, for each 15-minute time segment t, I solve for the numbers of usages r_{it} for each route $i \in \{1, \ldots, n\}$ using the following quadratic integer program (QIP):

$$\min_{\mathbf{r}_t, \mathbf{s}_t^{CV}, \mathbf{s}_t^{LD}} \|\mathbf{s}_t^{CV}\|_2^2 + \|\mathbf{s}_t^{LD}\|_2^2 + \lambda \sum_{i \in \text{nonfringe}} r_{it} \tag{5.1a}$$

$$\text{s.t. } \mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{CV} \geq \alpha_{LB}^{CV} \mathbf{f}_t^{CV} \tag{5.1b}$$

$$\mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{CV} \leq \alpha_{UB}^{CV} \mathbf{f}_t^{CV} \tag{5.1c}$$

$$\mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{LD} \geq \alpha_{LB}^{LD} \mathbf{f}_t^{LD} \tag{5.1d}$$

$$\mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{LD} \leq \alpha_{UB}^{LD} \mathbf{f}_t^{LD} \tag{5.1e}$$

$$\mathbf{r}_t \in \left(\mathbb{Z}^{\geq 0}\right)^n,$$

where $\mathbf{A} \in \{0,1\}^{n \times m}$ is a binary matrix denoting which counting locations are used by routes: $\mathbf{A}_{ij}$ is 1 if route i passes counting location j, and is 0 otherwise, such that $\mathbf{A}\mathbf{r}_t$ gives the number of times the generated routes collectively pass each counting location $j \in \{1, \ldots, m\}$; $\mathbf{s}_t^{CV}, \mathbf{s}_t^{LD} \in \mathbb{R}^m$ are slack variables that represent the error between the generated routes' counts $\mathbf{A}\mathbf{r}_t$ and the actual counts $\mathbf{f}_t^{CV}$ and $\mathbf{f}_t^{LD}$; nonfringe is the set of indices for routes where the start or the end of the route are interior edges in the road network; and λ is a hyperparameter for weighting the objective function penalty for these routes.

In the QIP, constraints (5.1b) and (5.1c) specify that $\mathbf{A}\mathbf{r}_t$ should lie within a probable range of counts extrapolated from computer vision counts. The lower bound $\alpha_{LB}^{CV} \mathbf{f}_t^{CV}$ assumes that

computer vision is overcounting, and the upper bound $\alpha_{UB}^{CV}\mathbf{f}_t^{CV}$ assumes that it is undercounting. The sum-of-squares of the error $\|\mathbf{s}_t^{CV}\|_2^2$ is minimised in the objective function (5.1a). The two following constraints, (5.1d) and (5.1e), are analogous constraints for loop detector counts. Again, the error $\|\mathbf{s}_t^{LD}\|_2^2$ is minimised in the objective. Finally, the objective minimises the number of nonfringe routes, which are rare under my assumptions.

Once I have obtained the solution $\mathbf{r}$, I assume that the set of routes corresponding to this solution are uniformly distributed within the 15-minute time segment indexed as t.

5.3.3 LLM-Based Simulation Refinement from Natural Language Feedback

The QIP (5.1) is fundamentally underconstrained. In particular, most municipalities do not install camera detectors at every intersection, meaning that my computer vision method (Section 5.3.1) cannot be applied to the entire road network. For counting locations where only loop detectors are available, the generated counts could lie anywhere between the lower bound (constraints (5.1b) and (5.1d)) and the upper bounds (constraints (5.1c) and (5.1e)). Not all possible ways of assigning routes to match these counts are equally realistic. The domain knowledge of stakeholders, such as traffic engineers, can be leveraged to ensure that the traffic simulation is aligned with downstream use cases such as traffic analysis. Yet, without experience in optimization, it is difficult for these stakeholders to directly modify QIP (5.1) to align with their intuition.

My problem formulation is as follows. There are K pieces of structured natural language feedback $p^{(k)} = (t^{(k)}, j^{(k)}, l^{(k)})$, where each one consists of a time, intersection, and a natural language description of what corrections should be made to the simulated traffic state at this intersection. There is also code which solves the original problem $\mathrm{QP}^{(0)}$, and the route counts $\mathbf{Ar}^{(0)}$ obtained from solving $\mathrm{QP}^{(0)}$. The objective is to produce an updated problem $\mathrm{QP}^{(k)}$, which has been modified so that it will produce a new route set $\mathbf{r}^{(k)}$ that addresses $p^{(k)}$. I solve this problem by prompting an LLM with a prompt containing $(p^{(k)}, \mathrm{QP}^{(k-1)}, \mathbf{Ar}^{(k-1)})$, and leveraging its code generation capabilities to generate $\mathrm{QP}^{(k)}$.

Notably, I do not provide the LLM with any handcrafted information beyond the time segment $t^{(k)}$ and intersection $j^{(k)}$ that the feedback is targeted at; what is already available from $\mathrm{QP}^{(k)}$; and a list of intersections and main roads. Based on the set of route counts $\mathbf{Ar}^{(k-1)}$ from the previous simulation, the LLM must automatically extract concrete, quantitative constraints that are aligned with the qualitative feedback. To solve this task, I prompt the LLM with a chain of thought [401]:

(1) Extract the relevant counts by formulating a command to call a `get_counts` tool, which retrieves the previous counts $\mathbf{Ar}_{jt}^{(k-1)}$ for a particular location and time segment (j, t)

(2) Write a constraint corresponding to the feedback $p^{(k)}$, using the counts from the previous step to make subjective judgments on how to set coefficients

(3) Translate this constraint to Python code for the package `cvxpy` 1.5.3 [86]

(4) For the time segment $t^{(k)}$ specified in the feedback, add this constraint to the optimization function while minimally modifying the rest of the code

(5) For the adjacent time segments $(t^{(k)}-1, t^{(k)}+1)$, add relaxed versions of the same constraint to ensure temporal continuity

I use this LLM within an iterative simulation refinement framework, similar to that of Behari et al. [31]. For each of K pieces of feedback, I sample a batch of B programs generated by the LLM. Then, I implement a rapid verification procedure to filter out incorrect programs: I first evaluate the program to ensure it represents syntactically correct Python, and then attempt to solve the new QIP for the time segment $t^{(k)}$. After discarding programs that fail this verification, I arbitrarily choose one program as $\mathrm{QP}^{(k)}$. In the next iteration, I prompt the LLM to directly modify $\mathrm{QP}^{(k)}$ to produce $\mathrm{QP}^{(k+1)}$. After KB generations, I obtain a single program $\mathrm{QP}^{(K)}$, which I execute for all timesteps t to obtain the final solution $\mathbf{r}^{(K)}$ and a corresponding simulation.

5.4 Simulation Results: Strongsville, Ohio

I applied my simulation generation pipeline (Section 5.3) to a large road network from the city of Strongsville, Ohio. The Strongsville road network experiences heavy through traffic due to its connection to two interstates, I-71 and I-80; the ramps of these interstates respectively connect to two intersecting arterials, SR 82 (Royalton Road) and US 42 (Pearl Road). On SR 82, the traffic volume of the road has exceeded its designed capacity, leading to the implementation of various countermeasures to improve throughput [100]. As part of these countermeasures, Strongsville installed an adaptive traffic signal control system on SR 82 and US 42.

This system uses three types of traffic detectors. (1) Camera detectors are used for the main roads at each intersection (i.e. along SR 82 and US 42) and on some side roads. (2) Loop detectors are used for most side roads and some turning movements. (3) Radar detectors are used for detection upstream and downstream of intersections. As my goal is to match the traffic state at the intersections themselves, I do not consider data from Strongsville's radar detectors.

My simulation, as shown in Figure 5.3, covers the intersections along SR 82 and US 42 for which the city has installed adaptive signal control. I first converted OpenStreetMap data to a SUMO [10] road network. Next, on Friday, September 6, 2024, I captured 24 hours of footage from 74 out of 86 counting locations where AutoScope detectors are installed. I imputed missing counts using those from AutoScope and loop detectors. After I applied my pipeline, I randomly assigned vehicles to different vehicle classes [404], following a survey conducted by the Ohio DOT in September 2022. Finally, I implemented the traffic signal patterns that were in use.

In the rest of this section, I evaluate the accuracy of my pipeline steps for this simulation to address my research questions from Section 5.1.

5.4.1 RQ1: Accuracy of Vehicle Counting

To evaluate the accuracy of my vehicle counting method (Section 5.3.1), I manually counted vehicular traffic from camera detector footage. As doing so would be infeasible for the entire simulation, I selected footage from four different intersections that are important to stakeholders (from the south, centre, east, and north of the road network), and two one-hour time segments (12 pm, an off-peak hour, and 5 pm, a peak hour) for each intersection. Figure 5.2 shows a screenshot from one of these pieces of footage.

Figure 5.3: Screenshot of traffic simulation of Strongsville, Ohio. Counts are available for 36 intersections, either derived from computer vision on raw camera detector footage (approaches in blue) or from loop/AutoScope detectors (approaches in orange). Each intersection is labelled with its number.

In Figure 5.4, I compare the performance of counts generated by my method and by AutoScope to the ground truth from manual counting. My evaluation suggests that my computer vision method was generally able to faithfully capture the traffic state of Strongsville. For all 16 of the footage excerpts that I and another co-author manually counted, my method had an error of less than 2 vehicles per minute (120 vehicles per hour). I used the same set of hyperparameters for preprocessing and detection/tracking across all counting locations; I believe that tuning these hyperparameters for individual counting locations could yield further gains.

Meanwhile, AutoScope exhibited a persistent pattern of undercounting across all of the approaches that I manually counted. In fact, it had an error of *more* than 3 vehicles per minute (180 vehicles per hour) for 14 out of 16 footage excerpts, with the exception being intersection 17's northbound approach (where I observed that lane switching resulted in duplicated actuations). The primary cause of this undercounting was the continuous actuation of detection zones by consecutive vehicles, as I discussed in Section 2.2.2. Consistent with this, AutoScope generally performed better under intermittent traffic during the 12 pm time segment, but its performance degraded under increased traffic levels during the 5 pm time segment.

An ablation experiment indicated that the preprocessing filters I applied to the footage had a significant impact on the accuracy of my vehicle counting method. The original footage included green detector actuation overlays that disrupted YOLO's ability to detect vehicles (Figure 5.2a).

Table 5.1: Comparison of vehicle counts obtained from my method (CV) and AutoScope (AS) with manual counting.

	8 — Pearl Rd & Echo Rd				17 — Pearl Rd & Business Entrance			
	Northbound		Southbound		Northbound		Southbound	
	12:00	17:00	12:00	17:00	12:00	17:00	12:00	17:00
Manual	823	959	781	1041	1042	952	1106	1329
CV	872	905	839	1071	1065	950	1114	1322
AS	582	637	594	688	1036	1026	647	895
	33 — Royalton Rd & I-71 Ramps				52 — Pearl Rd & Sprague Rd			
	Eastbound		Westbound		Northbound		Southbound	
	12:00	17:00	12:00	17:00	12:00	17:00	12:00	17:00
Manual	1889	2088	1059	726	979	1048	1166	1441
CV	1779	1962	1089	801	1096	1178	1191	1553
AS	1405	1406	587	220	442	578	853	960

Because these overlays overlapped with the position of the stop bar, this resulted in a failure to count any vehicles for some counting locations. Replacing the green pixels with black successfully remedied this issue. Frame-blending motion interpolation also mitigated the issue of BoT-SORT failing to associate vehicle tracks before and after dropped frames.

I used these results to estimate bounds for the ratios between my computer vision counts and the ground truth: $\alpha_{\text{LB}}^{CV} = 0.94, \alpha_{\text{UB}}^{CV} = 1.12$. As there is insufficient overlap between manual and loop detector counts, I extrapolated bounds for loop detector counts from the ratio between computer vision and loop detector counts. These bounds are loose due to error in the loop detectors:

$$\alpha_{\text{LB}}^{LD} = \alpha_{\text{LB}}^{CV} \min_t \min_{j \in CV \cap LD} \frac{f_{jt}^{CV}}{f_{jt}^{LD}} = 0.02, \alpha_{\text{UB}}^{LD} = \alpha_{\text{UB}}^{CV} \max_t \max_{j \in CV \cap LD} \frac{f_{jt}^{CV}}{f_{jt}^{LD}} = 19.06.$$

5.4.2 RQ2: Accuracy of Generated Simulation

Next, I solved the QIP (5.1) to generate a set of routes consistent with these counts (Section 5.3.2). From initial testing, I found that the objective function's value was mainly determined by the computer vision error $\|\mathbf{s}_t^{CV}\|_2^2$, which were two orders of magnitude larger than the loop detector error $\|\mathbf{s}_t^{LD}\|_2^2$ and the fringe route penalty $\sum_{i \in \text{nonfringe}} r_{it}$. To better balance the objective function, I chose to set the hyperparameter $\lambda = 10$. I obtained a simulation with a total volume of 182 230 vehicles over 24 hours. Among these vehicles, 72.64% had routes that started and ended on the fringes of the road network.

In Figure 5.5, I focus on the accuracy of my simulation for the time interval between sunrise (6:59 am) and sunset (7:51 pm) on September 6, 2024. For the average counting location, the

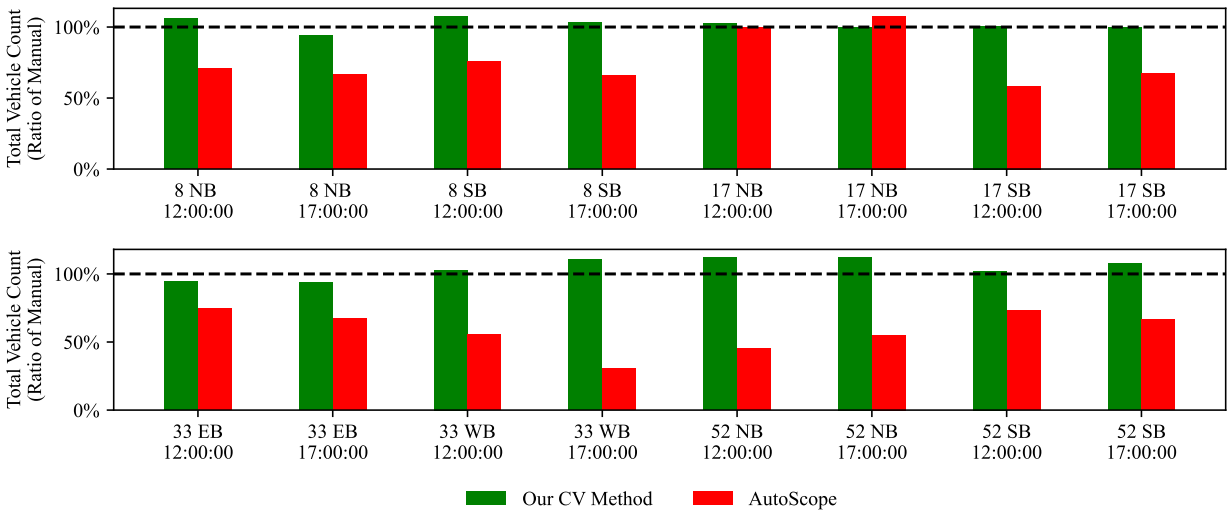


Figure 5.4: Plot of counts from my computer vision method (green) and AutoScope (red), as ratios relative to manual counts. If a method perfectly matches the manual counts, it has a ratio of 100%.

simulation was largely accurate to my computer vision counts f_j^{CV}, with no overflow or underflow. This can be attributed to the relatively narrow range of $[\alpha_{\text{LB}}^{CV}, \alpha_{\text{UB}}^{CV}]$. However, for many time periods, there are some counting locations where the simulation has a substantial overflow relative to the computer vision counts. I attribute these to counting locations with few vehicles where the traffic flow is inconsistent. Violating the expected range of these counts results in a small penalty compared to the rest of the objective function.

Meanwhile, the simulation matched the loop detector counts f_j^{LD} less closely; on average, the simulation's counts were approximately double that of the loop detectors. There was a considerable amount of both underflow and overflow for all time segments, but this is expected given the larger range of $[\alpha_{\text{LB}}^{LD}, \alpha_{\text{UB}}^{LD}]$. The underconstrained nature of the problem points to a need for stakeholder-driven refinement.

5.4.3 RQ3: Accuracy of LLM-Generated Constraints

As my LLM for simulation refinement (Section 5.3.3), I used Claude 3.7 Sonnet. An earlier iteration of this model achieved state-of-the-art performance on code generation benchmarks [463, 11]. I performed two rounds of evaluation: one on synthetic feedback (where a ground truth exists for constraint correctness), and one on real stakeholder feedback (where there is no ground truth). To increase diversity, I generated $B = 10$ programs for each piece of feedback with a temperature of 0.8. My evaluation focused on two criteria: *syntactic correctness*, whether the program is successfully evaluated as Python code; and *feasibility*, whether the program has a feasible solution.

First, I randomly generated $K = 20$ pieces of structured feedback in the form of (intersection, direction, approach, increase/decrease) tuples. I used Claude to rephrase this feedback to match the style of stakeholder feedback. For this feedback, I also assessed *semantic correctness*, by

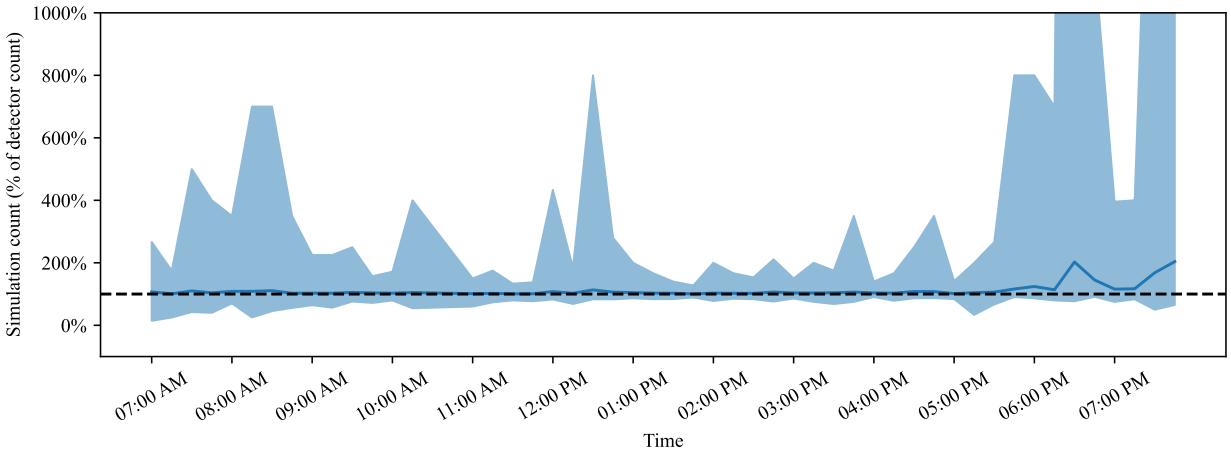


Figure 5.5: Plot of ratio between simulated counts and counts from my computer vision method. A simulation that is perfectly calibrated to the counts has a ratio of 100%. For each time segment, the solid line represents the mean across counting locations, while the shaded region is the range.

checking whether the generated volume in the updated simulation was indeed greater than/less than the original simulation. The LLM was always able to generate valid Python code, giving a syntactic correctness rate of **100%**. The feasibility rate was **87%**. Not all generated programs were feasible due to two issues in the added constraints: (1) they included a slack variable term s_t, which conflicted with the slack variable constraints from the original optimization problem, or (2) they were formulated in terms of computer vision counts f_t^{CV}, which were not always available. I found that the use of the `get_counts` tool was important to prevent hallucination of counts. Lastly, the semantic correctness rate was **87%** — whenever the generated program was feasible, the result was also correct. This gave me confidence in continuing to use this approach for feedback from stakeholders, after modifying the prompt to prevent the two aforementioned issues.

Second, I collected $K = 20$ pieces of feedback by presenting my simulation for the 5:00 pm time segment to a stakeholder familiar with Strongsville's traffic conditions. The LLM achieved a syntactic correctness rate of **100%** and a feasibility rate of **99%**. Most of this feedback indicated that my simulation was true to real life (particularly for intersections where traffic was based on my computer vision method). Thus, I focused on feedback for intersections where I could not apply my computer vision method.

Now, I demonstrate the capabilities of my framework with an example of the constraints generated by the LLM. Intersection 25 (US 42 & SR 82) was an intersection with camera detectors for which I was unable to capture footage. The stakeholder commented:

> I would expect to see more cars both eastbound and westbound at this time of day. Especially at this intersection at this time of day, I would expect every approach to max out. Each left turn, each through, main line, side street, I would expect it to be pretty packed at this time of day. There might be just a couple of lingering [vehicles], but I'd expect pretty much each queue to get its full allocated time.

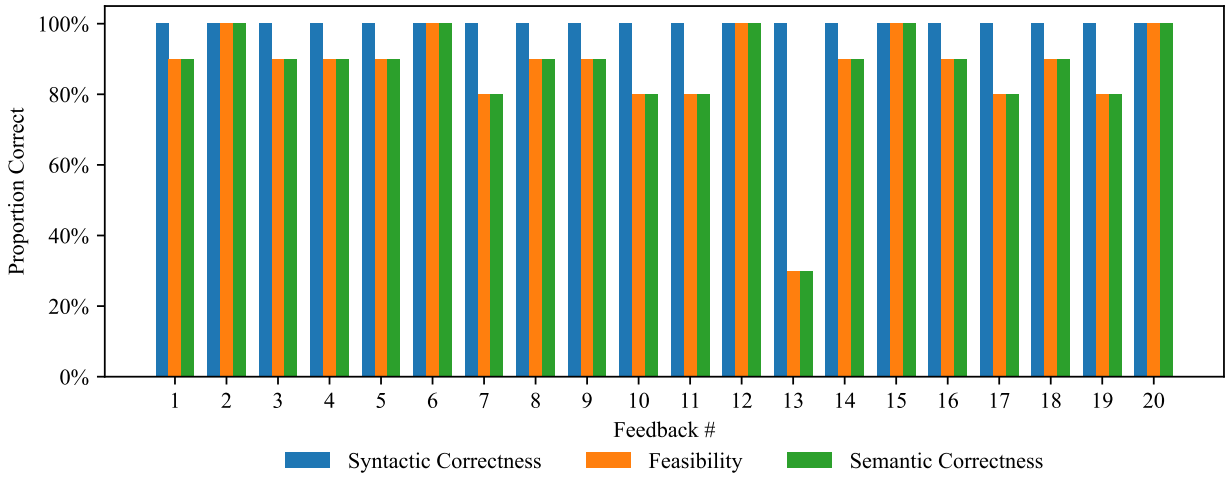


Figure 5.6: Evaluation of $B = 10$ optimization programs generated by Claude for $K = 20$ synthetically generated pieces of feedback. I used three criteria to evaluate the programs: syntactic correctness (perfect), feasibility (consistently high), and semantic correctness (less consistent).

The LLM queried the total eastbound, northbound, southbound, and westbound counts for the old simulation during the 5:00 pm time segment (respectively 119, 157, 281, and 334), as well as turn counts for the eastbound and westbound approaches. It then generated the following code:

```
# Add specific constraints for intersection 25 during 5:00 PM
# Indices for intersection 25 (EB, NB, SB, WB)
int25_indices = [40, 41, 42, 43]
# Apply higher traffic multiplier for 05:00 PM
if hour == "05:00 PM":
    # Add constraint to increase traffic at intersection 25
    for idx in int25_indices:
        # Ensure high traffic volume
        flow_constraints.append(cat_locs[:, idx, 0].T @
            route_count >= 300)

# Apply interpolated constraints for adjacent time periods
if hour == "04:45 PM" or hour == "05:15 PM":
    # Add slightly relaxed constraint for adjacent time periods
    for idx in int25_indices:
        # Slightly lower but still high traffic
        flow_constraints.append(cat_locs[:, idx, 0].T @
            route_count >= 250)
```

where `cat_locs` is the counting location matrix $\mathbf{A}$, and `route_count` is the solution $\mathbf{r}_t^{(k)}$.

From this snippet, I find that the LLM can correctly identify the indices of the counting locations; quantify the stakeholder's expectation of additional traffic by adding a minimum constraint

on the total flow; and add looser constraints to the adjacent time segments for continuity, all in valid Python code. Instead of hardcoding lower bounds, another program generated by the LLM applied multiplicative scaling factors to the counts from detector data f_t (`total_count_flow`):

```
# For the 5:00 PM time segment -
# increase traffic significantly based on feedback
if hour == "05:00 PM":
    for idx in int25_indices:
        flow_constraints.append((cat_locs[:, :, 0].T @
            route_count)[idx] >= total_count_flow[idx] * 1.5)
```

For other pieces of feedback, the LLM also applied scaling to the counts from the previous solution $\mathbf{Ar}_t^{(k-1)}$ (`prev_route_counts`). On intersection 6 (Pearl Rd & Royalton Rd), another intersection without available footage, the stakeholder commented:

> It's usually pretty packed on the main road. Pretty much from the first intersection of the corridor all the way. I think I would expect a little bit more traffic than that. That's another cross arterial through the city. I would expect a little more east and west traffic.

The generated code was

```
# Intersection 6 indices: 4 (EB), 5 (NB), 6 (SB), 7 (WB)
if hour == "05:00 PM":
    # Increase east-west traffic (EB and WB) by 40%
    flow_constraints.append((cat_locs[:, 4, 0].T @ route_count)
        >= 1.4 * (cat_locs[:, 4, 0].T @ prev_route_counts[h]))
    flow_constraints.append((cat_locs[:, 7, 0].T @ route_count)
        >= 1.4 * (cat_locs[:, 7, 0].T @ prev_route_counts[h]))

    # Increase north-south traffic (NB and SB) by 25%
    flow_constraints.append((cat_locs[:, 5, 0].T @ route_count)
        >= 1.25 * (cat_locs[:, 5, 0].T @ prev_route_counts[h]))
    flow_constraints.append((cat_locs[:, 6, 0].T @ route_count)
        >= 1.25 * (cat_locs[:, 6, 0].T @ prev_route_counts[h]))
```

My final simulation, after LLM adjustments, had a total volume of 191 361 vehicles over 24 hours. This increase can be in part attributed to increased volume at intersections from which computer vision counts could not be obtained.

5.5 Conclusion

In this chapter, I presented an end-to-end pipeline for generating a traffic simulation that includes three steps: computer vision-based vehicle counting, combinatorial optimization-based vehicle route generation, and LLM-based iterative simulation refinement from natural language feedback. I applied my pipeline to a high-traffic road network in the city of Strongsville, Ohio. Based on my evaluation results, I find that my pipeline is a demand modelling methodology that adheres

more faithfully to real-world traffic conditions than the approaches used in past work. Although I adapted my pipeline to the particularities of uncertainty and heterogeneity in Strongsville's detector infrastructure, my general framework can be easily generalised to other municipalities with comparable detection capabilities — which are becoming increasingly prevalent [353].

I envision that other sources of data could also be incorporated into my simulation framework. Road state reports (e.g. Waze), weather data, and business information can all be indicative of factors that impact traffic. As LLMs' capabilities improve, they hold promise for integrating data from these heterogeneous sources [51]. Furthermore, while my pipeline is offline, one line of future work is to convert it into a streaming pipeline capable of near-real-time use. Streaming capabilities would allow simulations to be updated based on live traffic. They would also enable the creation of an interactive interface where stakeholders can iterate on detection and optimization parameters using natural language feedback, while reviewing the results of their feedback instantaneously. This frontier of human-centred, AI-enabled possibilities can help traffic simulations to better reflect heterogeneity in real-world traffic conditions and to better serve their users.

5.6 Appendix

5.6.1 Full Claude Prompt

In the following prompt, the placeholders {intersection}, {int_idx}, and {feedback} are automatically filled in based on the feedback provided by the human stakeholder. {num_intersections}, {intersections_list}, and {main_roads_list} are filled in based on JSON-structured data, which can be customised to different deployment contexts.

Consider a road network with {num_intersections}. The following list shows the intersections, in the format "zero-based intersection index. (intersection number) - intersection name":
{intersections_list}
The main roads are:
{main_roads_list}

You are trying to generate a traffic simulation where vehicles follow routes that are consistent with observed counts.

For each intersection in the road network, you are given a count of how many vehicles enter the intersection in the eastbound, northbound, southbound, and westbound directions. Denote these counts as f_j for each of m counting locations j.

You are also given a set of n possible routes that vehicles can follow. Denote the number of times that route i is used as r_i. You are given a $n \times m$ matrix $\mathbf{A}$, where entry $\mathbf{A}_{ij}$ is 1 if route i passes counting location j, and is 0 otherwise.

For some counting locations j, you are also given specific counts for left turn vehicles and right turning vehicles. Denote these counts as $f_{LT,j}$ and $f_{RT,j}$ respectively. You are also given $n \times m$ matrices $\mathbf{A}_{LT}$ and $\mathbf{A}_{RT}$, where entries $\mathbf{A}_{LT,ij}$ and $\mathbf{A}_{RT,ij}$ are 1 if route i passes counting location j and respectively turns left or right, and are 0 otherwise.

The counting locations j are indexed as follows. Each of the intersections listed above occupies four consecutive entries for eastbound, northbound, southbound, and westbound.

You solve a quadratic program to find a solution for the n-dimensional variable $\mathbf{r}$.

This is the first requirement: For each counting location j, you want the sum of all routes i that pass through j to be within some range of the given counts f_j. Let $\mathbf{s}$ be an m-dimensional slack variable, and let α_{LB} (e.g. 0.94) and α_{UB} (e.g. 1.12) be lower and upper bounds on the given counts. This requirement can be represented as the constraints $\mathbf{Ar}+\mathbf{s} \geq \mathbf{f}\alpha_{LB}$, and $\mathbf{Ar}+\mathbf{s} \leq \mathbf{f}\alpha_{UB}$, and the objective function $\min_{\mathbf{s}} \sum_j s_j^2$.

This is the second requirement: For the counting locations j where turning vehicle counts are given, they should also be matched within some range of the given counts. Let $\mathbf{s}_{LT}$ and $\mathbf{s}_{RT}$ be m-dimensional slack variables, and let α_{LB} and α_{UB} be lower and upper bounds as before. This requirement can be represented as the constraints $\mathbf{A}_{LT}\mathbf{r} + \mathbf{s}_{LT} \geq \mathbf{f}_{LT}\alpha_{LB}$, $\mathbf{A}_{LT}\mathbf{r} + \mathbf{s}_{LT} \leq \mathbf{f}_{LT}\alpha_{UB}$, $\mathbf{A}_{RT}\mathbf{r} + \mathbf{s}_{RT} \geq \mathbf{f}_{RT}\alpha_{LB}$, and $\mathbf{A}_{RT}\mathbf{r} + \mathbf{s}_{RT} \leq \mathbf{f}_{RT}\alpha_{UB}$, and the objective function $\min_{s_LT, s_RT} \sum_j s_{LT,j}^2 + s_{RT,j}^2$.

This is the third requirement: Some of the routes begin and end in the middle of a road and are therefore less realistic. They can still be used to satisfy the counts, but as few as possible of them should be used. If "nonfringe" is the set of such unrealistic routes, this requirement can be represented as the objective function $\lambda \min_{\mathbf{r}} \sum_{i \in \text{nonfringe}} r_i$, where λ is a tunable parameter.

Altogether, this is the quadratic program that you solve:

$$\begin{aligned}
\min_{s, s_{LT}, s_{RT}} & \sum_j s_j^2 + s_{LT,j}^2 + s_{RT,j}^2 + \lambda \min_{\mathbf{r}} \sum_{i \in \text{nonfringe}} r_i \\
\text{subject to } & \mathbf{Ar} + \mathbf{s} \geq \mathbf{f}\alpha_{LB} \\
& \mathbf{Ar} + \mathbf{s} \leq \mathbf{f}\alpha_{UB} \\
& \mathbf{A}_{LT}\mathbf{r} + \mathbf{s}_{LT} \geq \mathbf{f}_{LT}\alpha_{LB} \\
& \mathbf{A}_{LT}\mathbf{r} + \mathbf{s}_{LT} \leq \mathbf{f}_{LT}\alpha_{UB} \\
& \mathbf{A}_{RT}\mathbf{r} + \mathbf{s}_{RT} \geq \mathbf{f}_{RT}\alpha_{LB} \\
& \mathbf{A}_{RT}\mathbf{r} + \mathbf{s}_{RT} \leq \mathbf{f}_{RT}\alpha_{UB} \\
& r \text{ is a vector of nonnegative integers}
\end{aligned}$$

You solve this problem once for every 15-minute interval over 24 hours.

Now, a traffic engineer has reviewed the simulation you generated from the solution. They have given some feedback on the traffic at the intersection with number {intersection}, which is at index {int_idx} in the list given above. This is their feedback for the "{time}" time segment: "{feedback}"

The attached Python function, `solve_routes`, solves this quadratic program using the Python library `cvxpy`. Modify this function to add one or more new constraints to the quadratic program so that it takes this feedback into account.

The inputs and constants in the function are as follows:

- `input_locs`: A 3D array for $\mathbf{A}_{\text{fringe}}$, the counting locations passed by the subset of realistic ("fringe") routes. The first dimension is the route, the second dimension is the counting location, and the third dimension is indexed as 0 = total flow, 1 = left turn flow, 2 = right turn flow. The indices of the counting locations follow the intersection numbering given previously.
- `eps_locs`: A 3D array for $\mathbf{A}_{\text{nonfringe}}$, the counting locations passed by the subset of unrealistic ("nonfringe") routes. The first dimension is the route, the second dimension is the counting location, and the third dimension is indexed as 0 = total flow, 1 = left turn flow, 2 = right turn flow.
- `cat_locs`: A 3D array formed by concatenating `input_locs` and `eps_locs`. The first dimension is the route, the second dimension is the counting location, and the third dimension is indexed as 0 = total flow, 1 = left turn flow, 2 = right turn flow.
- `primary_counts`: A 2D array for $\mathbf{f}$, the counts at each of the counting locations. These counts are more reliable and should be given priority in matching. Each row is a time segment and each column is a counting location.
- `secondary_counts`: A 2D array for $\mathbf{f}$, the counts at each of the counting locations. These counts are less reliable and do not need to be matched as closely. Each row is a time segment and each column is a counting location.
- `hour_labels`: An array of string labels following the format "(number with no leading zeroes):(two-digit number) (AM/PM)", e.g. "{time}".
- `scale_prim_lb`: A float for α_{LB}, denoting how much the traffic from more reliable primary counts should be scaled by as a lower bound, default 1.
- `scale_prim_ub`: A float for α_{UB}, denoting how much the traffic from more reliable primary counts should be scaled by as an upper bound, default 1.
- `scale_sec_lb`: A float for α_{LB}, denoting how much the traffic from less reliable secondary counts should be scaled by as a lower bound, default 1.
- `scale_sec_ub`: A float for α_{UB}, denoting how much the traffic from less reliable secondary counts should be scaled by as an upper bound, default 1.
- `eps_lambda`: A float, a hyperparameter used to weight the penalty for unrealistic ("nonfringe") routes.
- `time_lambda`: A float, a hyperparameter used to weight the squared loss between the solution from timestep t and timestep $t + 1$.
- `route_lambda`: A float, a hyperparameter used to weight the squared loss between the previous solution and the current solution.
- `start_time`: A datetime object representing a 24-hour timestamp of the day, before which secondary counts are used in place of primary counts (i.e. before sunrise).
- `end_time`: A datetime object representing a 24-hour timestamp of the day, after which secondary counts are used in place of primary counts (i.e. after sunset).

- `time_limit`: A float, a time limit for the quadratic program solver, default 60 seconds.
- `prev_route_counts`: A 2D array for $\mathbf{r}^{\text{old}}$, the solution from the previous iteration of the quadratic program. Each row is a time segment and each column is a route.
- `CONNECTIONS`: A dictionary, where each index is one of the intersections named above.
- `TURN_MAPS`: A dictionary, where the indices are "EB", "NB", "SB", and "WB" for the incoming directions at each intersection.

Let's think step by step.

1. Compute the counts of vehicles that entered intersection {intersection} from each direction under the previous simulation during the "{time}" time segment. The total counts ($\mathbf{A}\mathbf{r}$), left-turn counts ($\mathbf{A}_{LT}\mathbf{r}$), and right-turn counts ($\mathbf{A}_{RT}\mathbf{r}$) for the previous solution $\mathbf{r}^{\text{old}}$ are accessible through the `get_counts` tool. DO NOT use the tool any more than the minimum number of times that is necessary to answer this question.
    a. Determine what counting location, time segment, and count type flag to pass into the `get_counts` tool.
    b. Using the `get_counts` tool, determine how many vehicles entered intersection {intersection} during the "{time}" segment from the eastbound, northbound, southbound, and westbound directions.
    c. If the count for a particular direction is 0 for every time segment during the day, that means that the given direction does not exist at this intersection. You should not add a constraint for this particular direction at this intersection, or else the problem will be infeasible.
2. Based on the traffic engineer's feedback, determine if the current simulation is reflective of real-world traffic conditions at this intersection. If it is reflective, you should add a constraint to maintain the current level of traffic at this intersection. If it is not reflective, you should add a constraint to modify the current level of traffic at this intersection.
3. In mathematical notation, write out a constraint corresponding to the traffic engineer's feedback. This constraint may involve parameters with unspecified values, along with f_j and $\mathbf{A}_{ij}$. It should NOT involve the slack variables s_j, as the constraint may make the problem infeasible.
4. Use your best judgement to determine what values of the unspecified parameters in the constraint would best correspond to the traffic engineer's feedback.
5. Explain how your parameter values from step 4 are sensible, given the eastbound, northbound, southbound, and westbound traffic at intersection {intersection} that you calculated in step 2 for the previous simulation.
6. Determine the indices of counting locations (j) corresponding to intersection {intersection}.
    (a) What is the index of intersection {intersection} in the list of intersection names given above? This should be a 0-based index.
    (b) If each intersection occupies 4 counting locations, what are the counting locations for

intersection {intersection}? These should be 0-based indices. For example:

Intersection 4 is the first entry (index 0) in the list given above. It would occupy indices $j = 0 \times 4 + 0 = 0$, $0 \times 4 + 1 = 1$, $0 \times 4 + 2 = 2$, and $0 \times 4 + 3 = 3$.

Intersection 6 is the second entry (index 1) in the list given above. It would occupy indices $j = 1 \times 4 + 0 = 4$, $1 \times 4 + 1 = 5$, $1 \times 4 + 2 = 6$, and $1 \times 4 + 3 = 7$.

7. Translate the constraint from step 4 to a single line of `cvxpy` code.

   It should NOT involve the slack variables named `slack_prim`, `slack_lt_prim`, `slack_rt_prim`, `slack_sec`, `slack_lt_sec`, or `slack_rt_sec`, because otherwise the problem may be infeasible.

   It should also NOT involve the count variables named `primary_counts` and `secondary_counts`, because they may not exist. Instead, the constraint should be in terms of hardcoded numerical values wherever possible; failing that, it should be in terms of the previous solution `prev_route_counts`; and failing that, it should be in terms of `total_count_flow`, `lt_count_flow`, or `rt_count_flow`.

8. For the "{time}" time segment, add the line of code from step 7 to the list of constraints in the `solve_routes` function. You should modify the rest of the code as little as possible.

9. For the 15-minute time segments adjacent to the "{time}" time segment, add a similar constraint that interpolates between normal traffic and traffic that follows the traffic engineer's feedback. If possible, use the same constraint for both before and after. You should modify the rest of the code as little as possible.

10. Output the entirety of the modified `solve_routes` function. Do not skip any lines of code. Beyond the constraints you just added, you should modify the rest of the code as little as possible.

Chapter 6

Self Control

Imbuing Reinforcement Learning Algorithms for Coordinated Traffic Signal Control with Operational Constraints

Domain: Traffic signal control
Challenges: Assurance, coordination

6.1 Introduction

In Chapter 5, I created a realistic traffic simulation for a road network from the city of Strongsville, Ohio, and I studied how AI techniques can be used to integrate data from traffic detectors with inherent uncertainty and heterogeneity. This chapter focuses on the intended downstream application of my simulation: as a training and evaluation environment for traffic signal control (TSC) policies based on reinforcement learning (RL). As I outlined in Chapter 1, the goal of training such a policy should be to deploy it in a physical transportation system, so that it can improve the efficiency of traffic signals over existing plans from the perspective of various stakeholders. What characteristics, then, would make an RL-based signal plan trained using my simulation suitable for deployment with Strongsville's real-world infrastructure?

On one hand, signal plans should be *coordinated*. Transportation agencies aim to optimise traffic broadly throughout an entire road network, not just locally at intersections. The key system-level objectives of TSC, such as throughput and green progression [84], involve flow between individual intersections and thus can only be achieved by coordinating the behaviour of their controllers. State-of-the-art RL algorithms for TSC such as CoLight [397] do well at coordination, being able to replicate green progression and other desirable traffic patterns.

On the other hand, signal plans should also provide *assurances* to stakeholders. In practice, stakeholders impose various and constraints on how they would like signal plans to operate. Drivers seek predictability and consistency in traffic signals; they experience stress [170] and express complaints [40] at suboptimal signal patterns that differ from what they are used to. Traffic engineers also seek to understand and explain the impact of signal changes using their experience-

based mental models; black-box signal plans (such as those based on RL) hinder these goals [347]. For all of these stakeholders, *safety* is a paramount consideration that underlies their operational constraints, and it is one that cannot be compromised in exchange for better performance [430]. Misalignment in mental models often leads to crashes and other safety issues.

The city of Strongsville currently uses Econolite's Edaptive algorithm for adaptive signal control. As I discussed in Section 2.2.3, Edaptive applies heuristic optimisation methods to adjust base time-of-day patterns according to signal plan constraints. Based on discussions with the city's Traffic Management Centre and other relevant stakeholders, I learned that it is critical for deployed RL-based signal plans to follow the constraints and format of Edaptive's signal plans. An example of such a plan is shown in Figure 6.1). Every two cycles, Edaptive uses the past 15 minutes of traffic data to increase or decrease splits. Minimum and maximum green constraints are also enforced but not shown in this interface. Edaptive may also alter the cycle length and offsets every 3 runs; for simplicity, I consider cycle lengths and offsets to be fixed in this chapter.

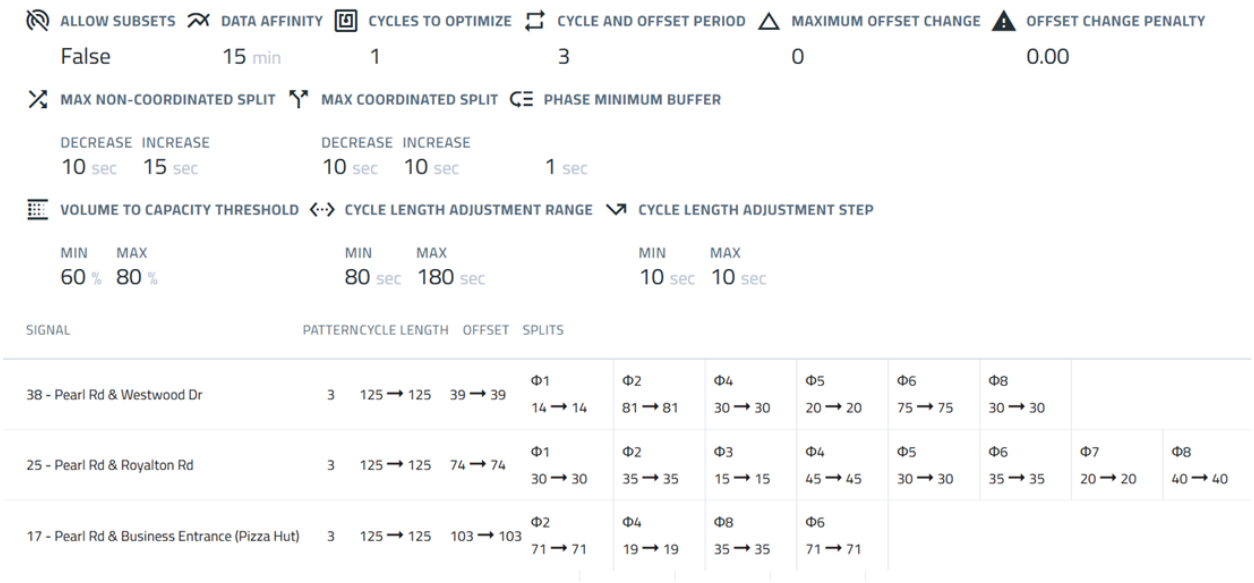


Figure 6.1: Edaptive configuration and output for the central US 42 corridor in Strongsville, Ohio, as of September 6, 2024 (the date corresponding to the simulation created in Chapter 5).
(Top) Constraints imposed on the Edaptive optimisation procedure.
(Bottom) Sample adjustment to signal plan offsets and splits made by Edaptive at 5:17 pm.

Edaptive's signal plans obey the following constraints, which I formally define in Section 6.3:

- **Minimum and maximum green**: These constraints on the duration of each phase ensure safety by giving road users from all approaches enough time to traverse an intersection [182]. Maximum green constraints also mitigate frustration from drivers stuck in traffic [170].
- **Fixed cycle length**: This constraint on the total duration of all phases in a cycle improves the predictability of signal plans for drivers, who use information about the time remaining in phases to determine their behaviour in response to phase changes [297].

- **Fixed phase sequence**: This constraint that requires a consistent ordering of phases within a cycle likewise improves the predictability of signal plans for drivers, and also mitigates losses in efficiency from discontinuous changes between different phases [137].
- **Smooth signal plan transition**: This constraint on the allowable change in splits between adjacent cycles both improves the verifiability of signal plans, and also mitigates losses in efficiency caused by transitions in splits between inconsistent plans [40].
- **Split predetermination**: This constraint that requires the entire sequence of splits for a cycle to be specified in advance improves the verifiability of signal plans for traffic engineers.

In this chapter, I introduce CycleLight, a suite of techniques that can be used to control the behaviour of signalling policies generated by state-of-the-art RL algorithms for TSC. This adds a dimension of assurance to algorithms that already exhibit strong coordination capabilities. First, I address the first four of these constraints using *action masking*, where phase-based acyclic policies are prevented from selecting invalid actions. Next, I address these constraints in conjunction with split predetermination using *action projection*, where split-based cyclic policies are projected back to safe sets through differentiable optimisation. To ensure that these cyclic policies achieve strong performance, I train them using *imitation learning* of performant acyclic policies.

For evaluation, I apply these techniques to the MPLight [54] and MA2C [64] algorithms, which I use to train signal control policies for the Strongsville simulation from Chapter 5. I assess how adding and removing each of the operational constraints above impacts both of the most important desiderata in TSC: *performance* and *safety*. To assess safety, I also implement proxy metrics of crash rates as novel reward functions for RL-based TSC.

6.2 Related Work

6.2.1 Safety Constraints in Reinforcement Learning

What does it mean for an RL agent's behaviour to be "safe"? Various notions of safety have been proposed based on different components of the RL problem formulation. In the most general form, a safety specification φ defines the sets of states, actions, or rewards that are safe or unsafe for an RL agent [8], such that $\varphi(s, a, r) = 1$ if the agent is safe and 0 if the agent is unsafe (which it should avoid) [185]. Different methods have been developed for each of these aspects of the RL problem formulation, although they are complementary to each other. In this chapter, I focus on safety in actions, but I also briefly discuss safety in states and safety in rewards.

For safety in actions, the safety specification defines a set of safe actions in each state, $\mathcal{A}_\varphi(s) = \{a \in \mathcal{A} \mid \varphi(s, a) = 1\}$. There are three broad categories of methods that guarantee the safety of actions taken by RL policies [185]:

- **Action masking** methods apply most directly to Q-learning and other RL algorithms for environments with discrete actions. For a deterministic Q-learning policy, the masked policy selects the safe action with the highest Q-value, $\tilde{a} = \text{argmax}_{a \in \mathcal{A}_\varphi(s)} Q(s, a)$. For a stochastic policy with action probabilities $\pi(a \mid s)$, the masked policy assigns probability 0 to unsafe actions and then renormalises the probabilities of the safe actions: $\tilde{\pi}(a \mid s) = \frac{\varphi(s,a)\pi(a|s)}{\sum_{a' \in \mathcal{A}_\varphi(s)} \pi(a'|s)}$

[33, 149, 383]. The latter approach was used to implement signal plan constraints traffic signal control by Müller and Sabatelli [264]. Unlike them, I extend action masking to account for cycle-level (not just phase-level) constraints involving many agents.

- **Action replacement** methods replace an unsafe action a having $\varphi(s, a) = 0$ with some fallback action $\tilde{a}$, which can be obtained either from sampling the safety set, as in *shielded RL* [8], or by using an alternative failsafe controller [24, 150, 327]. While Krasowski et al. [185] found that action replacement with sampling worked well on control tasks with continuous action spaces, I show in this chapter that the same is not true of multi-agent TSC. I do not consider failsafe controller methods, as no well-defined failsafe exists in TSC.
- **Action projection** methods are similar to action replacement methods, but also minimise a notion of distance between the replacement action $\hat{a}$ and the original action a:

$$\begin{aligned}\tilde{a} = \underset{a'}{\operatorname{argmin}}\ & \operatorname{dist}(a, a') \\ \text{s.t. } & \varphi(s, a') = 1\end{aligned}$$

  Two common families of approaches are based on model predictive control: *control barrier functions* (CBFs) [61, 403] and *safe set algorithms* (SSAs) [224, 454]. These approaches are rigorous, but also difficult to apply due to challenges in formulating RL as control problems [318]. In this chapter, I take a more direct optimisation approach to project vector-valued traffic signal control actions, and I also combine action projection with imitation learning.

For safety in states, the formulation is similar to safety in actions. There is a set of safe states in each of which *some* possible safe action can be taken, $\mathcal{S}_\varphi = \{s \in \mathcal{S} \mid \exists a \in \mathcal{A}, \varphi(s, a) = 1\}$. Due to the similar problem formulation, many action projection methods also apply to state safety [454]. Both CBFs and SSAs can be used to constrain actions in a way that provides theoretical guarantees about the overall state distributions encountered by agents, either asymptotically or during training. However, to provide these guarantees, these methods require perfect knowledge or specific structural properties of transition dynamics [454], which are not known in TSC. Therefore, I do not consider such global notions of state safety in this chapter.

For safety in rewards, a *cost function* is typically used to penalise unsafe behaviour. The most typical formulation is a *constrained MDP* $(\mathcal{S}, \mathcal{A}, P, R, \gamma, C, d)$, where $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the cost function, $\mathcal{C} = \sum_{t=0}^{T} \gamma^t C(s_t, \pi(s_t))$ is the *cost return*, and d is an upper bound on the cost return. The objective of a constrained MDP is to maximise the expected reward return while bounding the expected cost return [127], i.e.

$$\begin{aligned}\pi^* = \underset{\pi}{\operatorname{argmax}}\ & \mathbb{E}_{s_0 \sim \rho, s_{t+1} \sim P(s_t, \pi(s_t))} \mathcal{R} \\ \text{s.t. } & \mathbb{E}_{s_0 \sim \rho, s_{t+1} \sim P(s_t, \pi(s_t))} \mathcal{C} \leq d.\end{aligned}$$

Constrained MDPs are typically solved using *Lagrangian methods* [134, 298, 421], where the constrained MDP is converted to and solved as an unconstrained MDP. Specifically, the cost function is added to the reward function as a penalty and weighted adaptively by a Lagrange multiplier λ, giving $\operatorname{argmax}_\pi \min_\lambda(\mathcal{R} - \lambda\mathcal{C})$ as the equivalent unconstrained objective. While Lagrangian

methods are a more rigorous alternative to reward shaping techniques, the same challenges of appropriate definition and proper optimisation exist for cost functions as they do for reward functions. Huang and Ontañón [149] also found that the performance of action-based methods scales better than reward-based methods. Thus, I do not use the constrained MDP formulation in this chapter.

6.2.2 Safety Assessment in Traffic Signal Control

What does it mean for a traffic signal plan to be "safe"? The metric that is most directly of concern is the rate of crashes or *conflicts* (i.e. road user interactions that would lead to crashes if not averted) [117]. For safety reasons, it is not possible to deploy a plan and evaluate the crash rate in the real world, and crashes are rare events regardless. Therefore, virtually no physical data exists regarding how different signalling policies affect crash rates [309]. While crash prediction models exist, they have mostly been designed for freeways and other simpler, limited-access roadway designs [148]. Crash prediction models dedicated to arterial roadways have focused on indirect features such as speed, volume, and weather [429], and are of limited utility in signal plan design.

As I established in Chapter 4, traffic simulation provides an alternative means of systematically analysing the safety of signal plans. While simulations can be used to directly compute crash rates, a more efficient workflow is to use them to evaluate *safety performance functions*. These functions are regression models that rely on proxy metrics of crash rates [212], and are fit using large datasets from simulations or traffic monitoring systems. One of these is the US Federal Highway Administration's Surrogate Safety Assessment Model (SSAM) [117]. The most relevant application of SSAM was by Sabra et al. [309], who used it to model how different modifications to a Synchro-optimised signal plan's cycle length, offset, and splits impact crash rates.

Among other proxy metrics, SSAM uses the time to collision and the post-encroachment time of conflicts [60] to estimate crash rates. While such proxy metrics for crash rate analysis are practically significant, they are virtually unknown to researchers in RL for TSC [274]. I am aware of only two exceptions: Mirbakhsh and Azizi [254] evaluated the performance of RL on a single intersection with and without safety-based reward components, while Karbasi et al. [166] performed a similar evaluation to assess the impact of connected and automated vehicles on safety. In this chapter, I use the rate of constraint violations to evaluate safety for RL-based traffic signal plans. Unlike past works in the RL literature, I perform my evaluation using the high-fidelity, 36-agent simulation of Strongsville from Chapter 5.

6.3 Problem Formulation

In this chapter, I follow the problem setup of TSC from Section 2.2.1: a set of intersections in a road network are each controlled by one agent, and the goal of each agent is to take signalling actions for its intersection to optimise various signal performance measures. I thus also model TSC as a multi-agent RL problem using the *Markov game* formulation I introduced in Section 2.3.2.

Within this formulation, I assume that each timestep represents some atomic unit of real time, e.g. 5 seconds. Each intersection has a set of valid phases $\Phi_i = \{\phi_i^{(1)}, \ldots, \phi_i^{(P_i)}\}$. At timestep t, each intersection i is in phase $\phi_{i,t} \in \Phi_i$, and its agent i takes a signalling action $a_{i,t} = \Phi_{t+1} \in \Phi_i$.

(Henceforth, for simplicity, I drop the suffix i and consider each intersection separately.) In a given policy rollout, a phase ϕ may be signalled for K_ϕ blocks of contiguous timesteps, which can be considered analogously to *options* or other temporally extended action abstractions [16]. For each block $k \in \{1, \ldots, K_\phi\}$, the starting timestep is $t^{\text{start}}_{\phi,k}$, the ending timestep is $t^{\text{end}}_{\phi,k}$, and the duration of the block is the split for that phase, $t_{\phi,k} := t^{\text{end}}_{\phi,k} - t^{\text{start}}_{\phi,k}$.

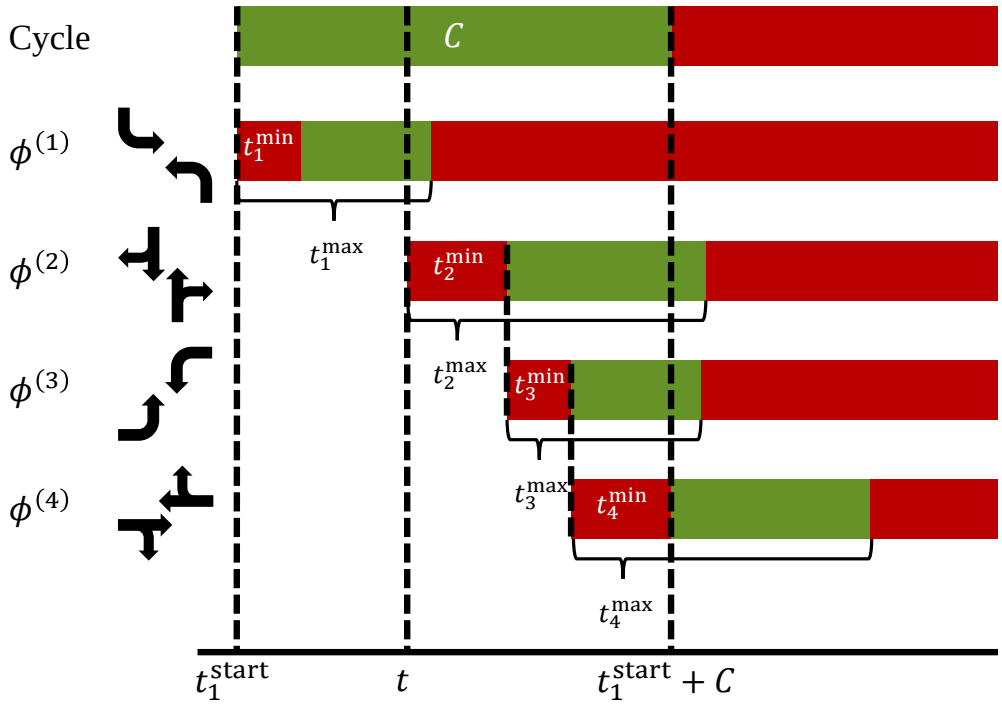


Figure 6.2: Diagram of signalling constraints for a four-phase intersection. For each phase, green denotes permissible durations, while red denotes prohibited durations. The intersection has an overall cycle length C, which the splits for all phases should sum to. Phases $\phi^{(1)}$ to $\phi^{(4)}$ each have a minimum duration, $t^{\min}_\phi$, and a maximum duration, $t^{\max}_\phi$. $\phi^{(1)}$ cannot be signalled up to its maximum duration; otherwise, the cycle is too short for the other phases' minimum durations.

Then, the constraints outlined in Section 6.1 can be formulated as follows:

- **Minimum and maximum green**. Given *minimum green times* $t^{\min}_\phi$ that lower bound the duration for each phase:

$$t_{\phi,k} \geq t^{\min}_\phi, \forall \phi \in \Phi, k \in \{1, \ldots, K_\phi\}. \tag{6.1}$$

  Similarly, given *maximum green times* $t^{\max}_\phi$ that upper bound phase durations:

$$t_{\phi,k} \leq t^{\max}_\phi, \forall \phi \in \Phi, k \in \{1, \ldots, K_\phi\}. \tag{6.2}$$

  In Figure 6.2, each phase $\phi^{(p)}$ is shown with a green range on the time axis, denoting its allowable duration under these two constraints. Any shorter, and it would violate the minimum green constraint; any longer, and it would violate the maximum green constraint.

- **Fixed cycle length**. Given a total *cycle length* C_k, the splits of all phases must sum to it:

$$\sum_{\phi} t_{\phi,k} = C_k, \forall k \in \{1, \ldots, K\}, \tag{6.3}$$

where $K = K_\phi, \forall \phi \in \Phi$ now represents the index of the cycle. C_k is not required to be constant across intersections and cycles, although this is often true in practice. Note that this constraint requires the RL policy to signal all phases $\phi \in \Phi$ exactly once during a cycle, although it does not constrain the order in which this occurs. Thus, there is an implicit constraint on the ordering of starting and ending timesteps for phases in adjacent cycles:

$$\max_{\phi \in \Phi} t^{\text{end}}_{\phi,k} < \min_{\phi \in \Phi} t^{\text{start}}_{\phi,k+1}, \forall k \in \{1, \ldots, K-1\}. \tag{6.4}$$

In Figure 6.2, the overall cycle length C is shown in green; all four phases, $\phi^{(1)}, \ldots, \phi^{(4)}$, must be signalled in this time. Beyond this duration, a new cycle (in red) must begin.

- **Fixed phase sequence**. Across all cycles $k \in \{1, \ldots, K\}, K = K_\phi, \forall \phi \in \Phi$, the start and end times of the phases always must occur in the same fixed order, $\{\phi^{(1)}, \ldots, \phi^{(P)}\}$:

$$\begin{aligned} &t^{\text{start}}_{1,k} < t^{\text{start}}_{2,k}, t^{\text{end}}_{1,k} < t^{\text{end}}_{2,k}, \\ &\forall \phi_1 < \phi_2 \in \Phi, k \in \{1, \ldots, K\}. \end{aligned} \tag{6.5}$$

- **Smooth split transition**. Between two temporally adjacent cycles, the splits of a given phase can change by no more than some threshold ϵ_k:

$$\max_{\phi \in \Phi} |t_{\phi,k} - t_{\phi,k+1}| \leq \epsilon_k, \forall k \in \{1, \ldots, K-1\}. \tag{6.6}$$

My formulation allows the threshold to vary over time to encapsulate two different types of constraints: (1) *maintaining* the same splits for multiple cycles (in which case $\epsilon_k = 0$), corresponding to the time between Edaptive's optimisation runs, and (2) *limiting* the increase or decrease of splits during an optimisation run. To simplify the formulation from that Edaptive (Figure 6.1), I assume the same bound for increases and decreases in splits.

- **Split predetermination**. Let the current cycle index of intersection i (i.e. the index of the last cycle that has begun in this intersection) be k_t. Instead of each action $\mathbf{a}_t$ only deciding the next phase ϕ_{t+1}, it should decide the splits for the entire next cycle $k_t + 1$:

$$\mathbf{a}_t = \{t_{1,k_t+1}, \ldots, t_{P,k_t+1}\}, \forall t \in \{1, \ldots, T\}. \tag{6.7}$$

6.4 CycleLight

How can these signal operation constraints be imposed on RL-based policies? In this section, I describe CycleLight, a generalisable suite of postprocessing techniques for signalling actions taken by RL policies. CycleLight can be combined with different RL algorithms for TSC to train policies that satisfy these operational constraints.

6.4.1 Action Masking

As I noted in Section 2.3.1, state-of-the-art RL algorithms for TSC are acyclic in that their actions consist only of the phase for the next timestep, i.e. $a_t = \phi_{t+1}$ [274]. For these algorithms with discrete action spaces, CycleLight uses action masking to implement all of the constraints defined in Section 6.3 except for split predetermination.

Based on these constraints, CycleLight uses action masking to the set of phases Φ_{t+1} that can be signalled next. Let the current phase be $\phi^{(p)}$. Which constraints are enforced at timestep t depends on the duration of the current phase so far, $t^{\text{curr}}_{p,k}$, and the duration of the current cycle k so far, $\sum_{\phi \in \Phi} t^{\text{curr}}_{\phi,k}$. The following cases are possible:

- **Phase cannot be changed**. If the minimum green constraint Equation (6.1) is enforced and $t^{\text{curr}}_{p,k} < t^{\min}_{p}$, then only the current phase can be signalled: $\Phi_{t+1} = \{\phi^{(p)}\}$. This can also occur if the split transition constraint Equation (6.6) is enforced, if the phase has not been signalled for long enough relative to the previous cycle: $t^{\text{curr}}_{p,k} < t_{p,k-1} - \epsilon_k$.
- **Phase can be changed**. The current phase has been signalled for at least its minimum green time. If both the phase ordering and phase sequence constraints (Equations (6.4) and (6.5)) are enforced, only the current phase or the next phase in the sequence is valid: $\Phi_{t+1} = \{\phi^{(p)}, \phi^{((p+1) \mod P)}\}$. If only the former is enforced, the current phase or any other phase not signalled yet in this cycle is valid: $\Phi_{t+1} = \{\phi \in \Phi \mid \phi = \phi^{(p)} \vee t^{\text{curr}}_{\phi,k_t} = 0\}$. Otherwise, any phase is valid: $\Phi_{t+1} = \Phi$.
- **Phase must be changed**. The current phase has been signalled for too long. Several constraint violations can lead to this case. For instance, if the maximum green constraint Equation (6.2) is enforced, then this case occurs if $t^{\text{curr}}_{p,k} > t^{\max}_{p}$. This can also occur if the split transition constraint Equation (6.6) is enforced, if the phase has not been signalled for too long relative to the previous cycle: $t^{\text{curr}}_{p,k} > t_{p,k-1} + \epsilon_k$.

  Lastly, Figure 6.2 illustrates a case where the cycle length constraint Equation (6.3) can lead to this case. Let $\Phi^{\emptyset}_{k} \subseteq \Phi$ be the subset of phases that has not been signalled in cycle k. If Equation (6.5) is enforced, then $\Phi^{\emptyset}_{k} = \{\phi^{((p+1) \mod P)}, \ldots, \phi^{(P)}\}$; otherwise, $\Phi^{\emptyset}_{k} = \{\phi \in \Phi \mid t^{\text{curr}}_{\phi,k_t} = 0\}$. Under Equation (6.3), every phase must be signalled in a cycle, and each phase must be signalled for its minimum green time. This constraint is violated if not enough time in the cycle is left for the minimum greens of the phases left:

$$C_i - \sum_{\phi \in \Phi - \Phi^{\emptyset}_{k}} t^{\text{curr}}_{\phi,k} < \sum_{\phi \in \Phi^{\emptyset}_{k}} t^{\min}_{\phi}.$$

  In Figure 6.2, $\phi^{(p)}$ is $\phi^{(1)}$. While $\phi^{(1)}$ has not been signalled up to its maximum time, there is not enough time in the cycle C to signal $\phi^{(2)}$ for $t^{\min}_{\phi^{(2)}}$, $\phi^{(3)}$ for $t^{\min}_{\phi^{(3)}}$, and $\phi^{(4)}$ for $t^{\min}_{\phi^{(4)}}$.

  If both the phase ordering and phase sequence constraints (Equations (6.4) and (6.5)) are enforced, only the next phase in the sequence is valid: $\Phi_{t+1} = \{\phi^{((p+1) \mod P)}\}$. If only the former is enforced, any other phase not signalled yet in this cycle is valid: $\Phi_{t+1} = \{\phi \in \Phi \mid t^{\text{curr}}_{\phi,k_t} = 0\}$. Otherwise, any phase is valid: $\Phi_{t+1} = \Phi$.

A phase that is valid to signal in the current timestep may still lead to constraint violations in future timesteps. For instance, in Figure 6.2, signalling $\phi^{(1)}$ up to time t is valid; however, if the

policy signals $\phi^{(2)}$ for more than $t^{\min}_{\phi^{(2)}}$, the cycle length constraint Equation (6.3) will still be violated. Therefore, CycleLight solves integer linear programs (ILPs) to find a subset of phases Φ^{ILP}_{t+1} which will definitely lead the acyclic policy to generate splits that satisfy the signalling constraints.

Specifically, assume that the policy has signalled up to phase $\phi^{(p)}$ at timestep t, it is currently in the kth cycle, and its chosen actions so far have induced splits $\mathbf{t}_k$ for this cycle. Suppose the policy signals for t_Δ (e.g. 5 seconds) at a time. To verify whether signalling $\phi^{(p)}$ for t_Δ additional seconds will lead to feasible splits, CycleLight solves the ILP

$$\min_{\hat{\mathbf{t}}} 1 \tag{6.8}$$

$$\text{s.t.} \sum_{p'=p}^{P} t_\Delta \hat{\mathbf{t}}^{(p')} = C - \sum_{p'=1}^{p-1} t_\Delta \mathbf{t}_k^{(p')} \tag{6.9}$$

$$t^{\min}_{p'} \leq t_\Delta \hat{\mathbf{t}}^{(p')} \leq t^{\max}_{p'}, \forall p' \in \{p, \dots, P\} \tag{6.10}$$

$$\left| t_\Delta \hat{\mathbf{t}}^{(p')} - \mathbf{t}_{k-1}^{(p')} \right| \leq \epsilon_k, \forall p' \in \{p, \dots, P\} \tag{6.11}$$

$$\hat{\mathbf{t}}^{(p)} = \mathbf{t}_k^{(p)} + t_\Delta \tag{6.12}$$

$$\hat{\mathbf{t}} \in (\mathbb{Z}^+)^P, \tag{6.13}$$

where $\hat{\mathbf{t}}$ are the splits for phases $\phi^{(p)}, \dots, \phi^{(P)}$ divided by t_Δ; $\mathbf{t}_k$ are the undivided splits induced by the acyclic policy *so far* in the current cycle; and $\mathbf{t}_{k-1}$ are the splits induced in the previous cycle. Equation (6.9) satisfies the cycle length constraint (6.3); Equation (6.10) satisfies the phase duration constraints (6.1) and (6.2); Equation (6.11) satisfies the split transition constraint (6.6); Equation (6.12) constrains the split for the current phase $\Phi^{(p)}$ to be equal to its total duration if it were to be signalled; and Equation (6.13) constrains all of the phase times to be multiples of t_Δ. Verifying whether $\phi^{((p+1) \mod P)}$ is feasible involves a similar ILP, except the phases are divided into $\{1, \dots, p\}$ and $\{p+1, \dots, P\}$, and Equation (6.12) is replaced by $\hat{\mathbf{t}}^{((p+1) \mod P)} = t_\Delta$.

This ILP is similar to the LP solved by my schedule recommendation tool in Chapter 3 to reoptimise schedules while fixing existing shifts. Note that, as a simplifying assumption, I assume that the cycle length is a perfect multiple of the timestep increment, $C \mod t_\Delta = 0$. If not, I allow the last phase to have arbitrary length to satisfy the cycle length requirement: $\hat{\mathbf{t}}^{(P)} \in \mathbb{R}$.

For a deterministic policy, CycleLight selects the highest-weighted (e.g. by Q-value) valid phase from the ILP-filtered set Φ^{ILP}_{t+1} as the action for timestep $t+1$. For a stochastic policy, given action probabilities for each phase $\pi(a = \phi \mid s)$, CycleLight masks out invalid phases and renormalises the probabilities of the remaining valid phases, which is equivalent to applying Dropout deterministically to the policy's logits [185]:

$$\tilde{\pi}(a = \phi \mid s) = \frac{\mathbb{1}[\phi \in \Phi_{t+1}] \pi(a = \phi \mid s)}{\sum_{\phi' \in \Phi_{t+1}} \pi(a = \phi' \mid s)}.$$

It then samples from $\tilde{\pi}$ to produce the action for timestep $t+1$.

6.4.2 Imitation Learning

Cyclic RL algorithms for TSC, which output the splits for an entire cycle at once, satisfy the cycle length and predetermination constraints — Equations (6.3), (6.4) and (6.7) — by construction. Policies from these algorithms output logits $\sigma^{(p)}$ for each phase p, which are normalised into a distribution over the cycle time. Given a fixed cycle length C, for cycle k of intersection i, such a policy would output a length-P vector of the form

$$\mathbf{a}_k = \left\{ \frac{C\sigma^{(1)}}{\sum_{p=1}^{P} \sigma^{(p)}}, \ldots, \frac{C\sigma^{(P)}}{\sum_{p=1}^{P} \sigma^{(p)}} \right\}.$$

If these splits are always executed in the order $\{\phi^{(1)}, \ldots, \phi^{(P)}\}$, Equation (6.5) is also satisfied. At the start time of cycle k, $t_{\text{start}} = t_{1,k}^{\text{start}}$, the policy takes action $\mathbf{a}_t = \mathbf{a}_k$. Until the end of the cycle at $t_{\text{end}} = t_{\text{start}} + C$, the policy takes no action.

Due to the larger parameter space and the reduced flexibility in actions, cyclic RL algorithms are likely to be less performant than their acyclic counterparts given the same amount of training time. Therefore, cyclic RL trades off a loss in performance for gains in interpretability and safety. However, I propose to use *imitation learning* to make cyclic RL more performant.

In imitation learning, a *student* or *learner* policy $\hat{\pi}$ is trained based on demonstrations from a *teacher* or *expert* policy π^* [205]. The goal is to minimise some loss between the student and the teacher, $J(\hat{\pi}) - J(\pi^*)$, so that the student achieves similar performance (e.g. in terms of returns) to the teacher [290]. For the TSC setting, the student $\hat{\pi}$ is a cyclic policy, and the teacher π^* is an acyclic policy trained with all relevant signalling constraints imposed.

To train each intersection agent's cyclic policy $\hat{\pi}$ with split actions $\mathbf{a}_k$, I backpropagate a reward $r_t = R(s_t, \mathbf{a}_k)$ for each cycle k after the action $\mathbf{a}_k$ has been fully executed. The agent's loss function $\mathcal{L}$ consists of three components. In addition to the base RL loss $\mathcal{L}_{\text{base}}$, I add a penalty of λ_{viol} for each constraint from Equation (6.1) to Equation (6.6) that is violated by the cyclic policy's splits $\mathbf{a}_k$, as well a Euclidean distance-based imitation learning loss $\mathcal{L}_{\text{dist}}$:

$$\begin{aligned}
\mathcal{L}(s_t, \mathbf{a}_k) =& \mathcal{L}_{\text{base}}(s_t, \mathbf{a}_k) + \lambda_{\text{viol}} \mathcal{L}_{\text{viol}}(s_t, \mathbf{a}_k) + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}}(s_t, \mathbf{a}_k) \\
=& \mathcal{L}_{\text{base}}(s_t, \mathbf{a}_k) + \lambda_{\text{viol}} \Bigg[\left(\sum_{p=1}^{P} \mathbf{a}_k^{(p)} - C \right)^2 + \sum_{p=1}^{P} \left(\min(0, t_p^{\min} - \mathbf{a}_k^{(p)}) \right) \\
&+ \sum_{p=1}^{P} \left(\min(0, \mathbf{a}_k^{(p)} - t_p^{\max}) \right) + \sum_{p=1}^{P} \mathbb{1} \left[\left| \mathbf{a}_k^{(p)} - \mathbf{a}_{k-1}^{(p)} \right| > \epsilon_k \right] \left(\mathbf{a}_k^{(p)} - \mathbf{a}_{k-1}^{(p)} \right)^2 \Bigg] \\
&+ \lambda_{\text{dist}} \left\| \mathbf{a}_k - \hat{\mathbf{t}}_k \right\|_2^2,
\end{aligned} \tag{6.14}$$

where $\hat{\mathbf{t}}_k$ are the splits that would have been induced by the acyclic policy in the same cycle. This loss function penalises the cyclic policy for constraint violations, as well as based on how much the

its splits deviate from the splits that the acyclic policy executed given the exact same observations. As shown in Figure 6.3, the imitation learning loss is on the level of cycles: rather than at individual timesteps, the cyclic policy imitates the acyclic policy's behaviour over entire cycles.

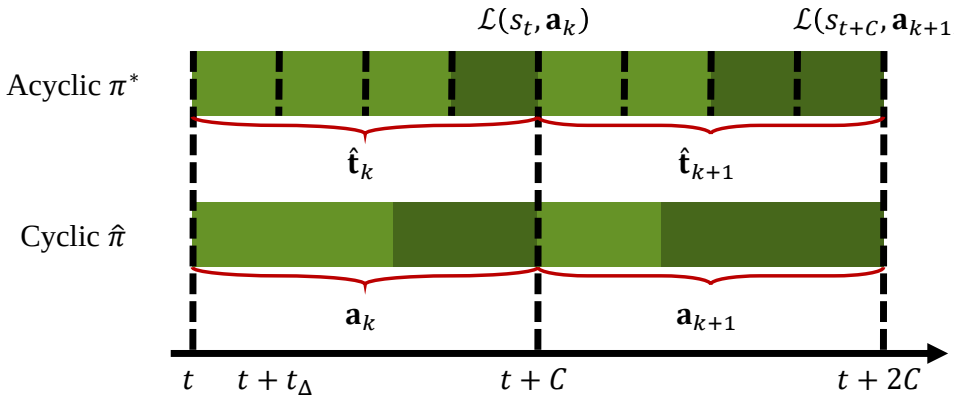


Figure 6.3: Diagram illustrating computation of imitation learning loss. Starting at time t, a constrained acyclic policy π^* and a cyclic policy $\hat{\pi}$ both signal phases over cycle length $C = 4t_\Delta$. $\hat{\pi}$ signals splits $\mathbf{a}_k$, while the phases signalled by π^* every t_Δ timesteps induce splits $\hat{\mathbf{t}}_k$. The loss for $\hat{\pi}$ is taken at the end of the cycle between $\mathbf{a}_k$ and $\hat{\mathbf{t}}_k$, using the state s_t at the start of the cycle.

6.4.3 Action Projection

By itself, imitation learning does not guarantee that a cyclic policy will follow all of the signalling constraints I consider. Although the penalty the loss R_{dist} incentivises the cyclic policy to behave similarly to the constrained acyclic policy, no hard constraints are placed on the actual cyclic policy's behaviour. To follow the remaining constraints, Equations (6.1), (6.2) and (6.6), CycleLight uses action projection by solving the following quadratic program (QP) to postprocess $\mathbf{a}_k$:

$$\hat{\mathbf{a}}_k = \underset{\hat{\mathbf{a}}}{\operatorname{argmin}} \|\hat{\mathbf{a}} - \mathbf{a}_k\|_2^2 \tag{6.15}$$

$$\text{s.t.} \sum_{p=1}^{P} \hat{\mathbf{a}}^{(p)} = C \tag{6.16}$$

$$t_p^{\min} \leq \hat{\mathbf{a}}^{(p)} \leq t_p^{\max}, \forall p \in \{1, \ldots, P\} \tag{6.17}$$

$$\left|\hat{\mathbf{a}}^{(p)} - \mathbf{a}_{k-1}^{(p)}\right| \leq \epsilon_k, \forall p \in \{1, \ldots, P\} \tag{6.18}$$

$$\hat{\mathbf{a}} \in \mathbb{R}^P,$$

where $\mathbf{a}_{k-1}$ is the action, i.e. splits, taken by the cyclic policy in the previous cycle. Equation (6.15) minimises the Euclidean distance between the original and projected splits, while Equations (6.16) to (6.18) enforce the cycle length, phase duration, and split transition constraints as before. These optimisation constraints are disabled if the corresponding constraints are not enforced.

One issue of a naïve split postprocessing approach is that policy gradients are lost during the projection step. Nevertheless, there is a straightforward way to obtain gradients from this optimisation problem. Using the differentiable optimisation package `cvxpylayers` [3], CycleLight is able to directly backpropagate through the action projection step. The use of `cvxpylayers` in CycleLight eliminates the need for reward shaping and hyperparameter tuning for the constraint violation penalty from Equation (6.14), and always produces a valid action.

6.5 Experiments

In my experimental evaluation of CycleLight, I apply it to train TSC policies for the Strongsville simulation from Chapter 5. By doing so, I seek to address the following research questions:

RESEARCH QUESTION 6.1. *Can CycleLight balance signalling performance with adherence to signalling constraints when it is applied to RL algorithms for TSC?*

RESEARCH QUESTION 6.2. *What is the effect of action masking, imitation learning, and action projection in CycleLight on the performance and safety of RL algorithms for TSC?*

RESEARCH QUESTION 6.3. *How do signalling constraints Equation* (6.1) *to (6.7), as implemented in CycleLight, impact the performance of RL algorithms for TSC?*

6.5.1 Environment

To create an interface between RL policies and the microscopic traffic simulator SUMO [10], I began by using the `sumo-rl` environment of Alegre et al. [6]. However, this environment was designed for acyclic policies and thus has a number of limitations. While `sumo-rl` implements minimum green (6.1) and maximum green (6.2) constraints, it enforces the same limits for all phases at a given intersection, whereas this is usually not true of signal plans in practice. Furthermore, it does not implement the execution of splits-based actions, and therefore provides no way to implement the cycle length (6.3), phase ordering (6.5), or split transition (6.6) constraints.

While retaining the basic framework of `sumo-rl`, I created a new wrapper environment for SUMO to address these limitations. From SUMO simulation files, the environment extracts individual yellow, red, and minimum and green durations for each phase, as well as the full set of signal plans and cycle lengths defined in the simulation. The environment also allows policies to execute actions as either individual phases (in which case they are signalled for $t_\Delta \geq 1$ timesteps) or as the full set of splits. It does so using an event queue: instead of each intersection's controller state being updated instantaneously and independently, actions executed by controllers are enqueued into a centralised queue. During every timestep representing 1 second of simulated time, the environment inspects the queue, and triggers the controllers to execute every signal state change event that occurs within that second. While the environment is capable of executing sub-second splits, I use an update granularity of 1 second to trade off simulation runtime and fidelity.

As shown in Figure 6.4, the environment does not enforce a one-to-one correspondence between simulation timesteps and agent actions. During every timestep, the environment updates the current reward of each agent so as to avoid biasing the average computed over the entire episode.

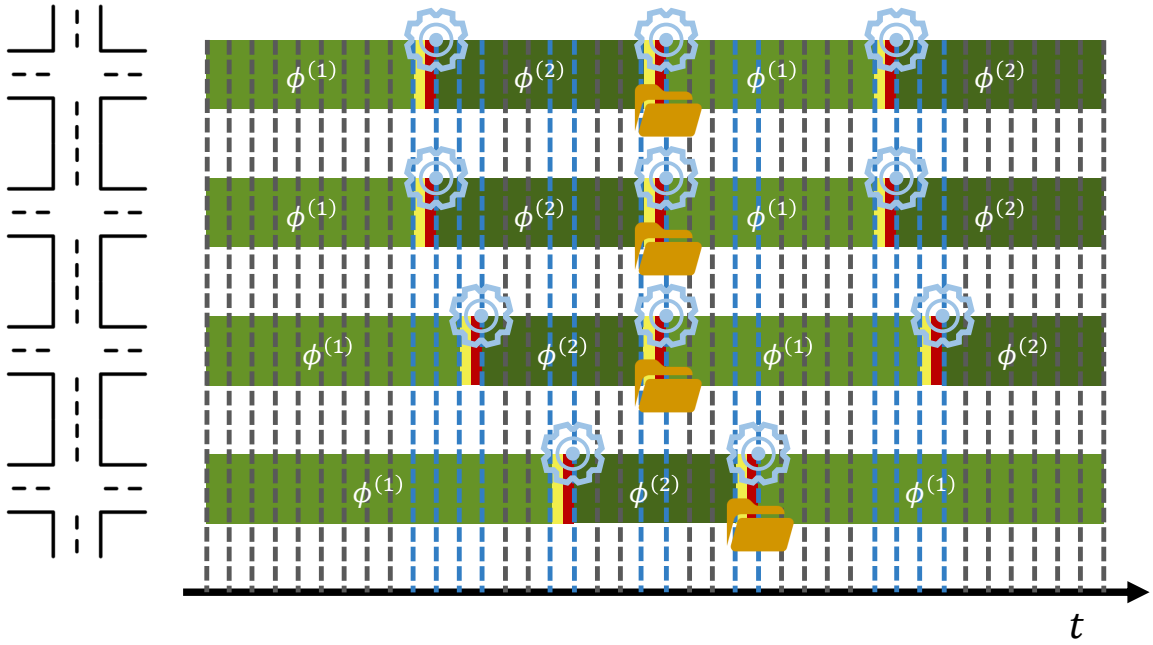


Figure 6.4: Diagram showing execution of a four-intersection traffic simulation in my event queue-based modification of the `sumo-rl` TSC environment, where each intersection has two phases and is controlled by a cyclic policy. The simulation updates the reward in each timestep (shown as dark dashed lines) *without* interfacing with intersection controllers, until it reaches a queued change in signal state for any intersection (shown as blue dashed lines). Blue gears denote these timesteps, at which a cyclic policy changes the intersection to a new phase. Yellow folders denote timesteps at which a cyclic policy adds an experience (based on the entire cycle) to its training dataset.

The environment only interacts with lower-level controllers to update the signal phase when indicated by the event queue. If the RL policy is acyclic, it queries the policy for new actions and adds an experience to its buffer. Otherwise, if the policy is cyclic, it only queries for a new action and adds a new experience once the entire sequence of enqueued splits has finished executing. In either case, the RL algorithm observes each action's cumulative effects on the reward during the training process: the observations stored into the buffer originate from the first timesteps of each action, while the rewards stored into the buffer originate from the last timesteps.

Using this RL environment, I run experiments on two traffic simulations.

- The first simulation is the *Cologne corridor* [377], a three-intersection environment representing a traffic corridor from the city of Cologne (Köln), Germany. It has a total volume of 4 494 vehicles in 7–8 am rush hour traffic. I used the implementation of this environment from the RESCO benchmark [14]. This implementation includes a default 90-second signal plan for each intersection, which has 3-second yellow change intervals but no red clearance intervals for each phase. It also constrains the minimum and maximum durations of each phase to 5 and 50 seconds for all intersections. I use this environment to iterate on the design of CycleLight, as it runs relatively quickly.

- The second simulation is the *Strongsville road network* from Chapter 5, using the final simulation of the 5–6 pm time segment after stakeholder feedback. For this environment, I replicated the city's signalling constraints as faithfully as possible. I extracted the phasing sequence for each intersection, along with their yellow, red, minimum green, and maximum green times, from the signal plans for US 42 (Pearl Road) and SR 82 (Royalton Road). However, to reduce the size and complexity of the action space, I removed overlap phases (e.g. in Figure 2.2, I combined the northbound and southbound left turn phases into one action, and straight through/right turn phases into another action), and I set the minimum and maximum green durations to the maxima over all combined phases.

  To set cycle lengths and split transition constraints, I replicated the base time-of-day patterns and optimisation constraints from the city's Edaptive configuration (Figure 6.1). In particular, I set each intersection's cycle length based on the signal plans that they executed during the simulated time segment: 130-second plans for US 42, and 112-second plans for SR 82. For the maximum split distance ϵ, I used the coordinated split limit of 10 seconds.

6.5.2 Algorithms and Baselines

I use CycleLight to impose signalling constraints on two state-of-the-art, acyclic RL algorithms for TSC. First, I use MPLight [54], a parameter-shared Q-learning algorithm that trains the symmetry-invariant FRAP model [455] to unify state embeddings across all intersections. Second, I use MA2C [64], an observation-sharing policy gradient algorithm that exchanges policy "fingerprints" between intersections that are adjacent in the road network. These represent two extremes of the tradeoff between scalability and coordination among RL algorithms for TSC (Section 2.3.4). The implementations for both algorithms were modified from the RESCO benchmark [14]. Additionally, I compare the performance of these algorithms to the default, fixed-time signal plan from the Cologne corridor and Strongsville network environments.

To implement imitation learning, I also design cyclic variants of these two algorithms, which I denote as Cycle-MPLight and Cycle-MA2C. MPLight policies are trained using a variant of the DQN algorithm [255], which does not support continuous actions. Instead, I use a continuous action implementation of the PPO algorithm [324] by Hao [140]. I also modify the FRAP model to output three heads for computing PPO's losses: a split head that outputs a size-P vector $\mathbf{a}$; a distribution head that outputs a Normal distribution $N(\mathbf{a}, e)$, for deriving the advantage from action log-probabilities; and a value head that outputs a scalar $V(s)$. MA2C is easier to modify, as it outputs a vector of logits that are normalised into a probability distribution $\pi(a \mid s)$. I multiply this distribution by the target cycle length to obtain splits $\mathbf{a} = C\pi(a \mid s)$.

All experiments were run in parallel on a server with 56 2.75GHz AMD EPYC 7453 processors and 252 GiB of RAM. RL algorithms were trained for 5 000 episodes with 600 timesteps (corresponding to 10 minutes of simulation time) in each episode. To solve the ILPs and QPs, I respectively use the packages `cvxpy` 1.5.3 [86] and `cvxpylayers` 0.1.6 [3], respectively using the Gurobi 12.0.2 [133] and ECOS 2.0.12 [89] solvers.

6.5.3 Evaluation Metrics

For the reward function and the primary performance evaluation metric of all algorithms, I use the *queue length*, taken as a sum across all intersections and as a mean across all timesteps in an episode. Accordingly, for all RL algorithms, the observations consist of the current phase and the queue lengths for each phase. To compute the queue length, I count the number of vehicles with speed less than 0.1 m/s in each incoming lane, and then group lanes into phases.

Additionally, I compute the *constraint violation rates* for each constraint enforced. For acyclic algorithms, these are mostly computed from the splits $\hat{\mathbf{t}}_k$ induced by the policy's actions in each cycle. The only exception is the phase ordering constraint, for which violations are tabulated every time the policy takes an action. For cyclic algorithms, these are computed with the splits $\mathbf{a}_k$ generated by the policy for each cycle. The violation rate for each intersection is computed as the *number of cycles* per episode (which is a constant in the simulation) in which a violation of any magnitude occurs. The overall violation rate is taken as a sum over intersections.

Lastly, I use SUMO's SSM devices [21] to compute two safety metrics that are used in SSAM and other common traffic analysis methods. In SUMO, vehicles with these virtual devices equipped monitor their trajectories for potential car-following, merging, or crossing conflicts between two vehicles, each of which is defined by a "conflict area". The two safety metrics are:

- *Time to collision*. This is the time for a vehicle to enter a conflict area if it does not change its behaviour, as given by the distance to the conflict area divided by the vehicle's speed.
- *Post-encroachment time*. This is the amount of time between when the first vehicle exits the conflict area and the second vehicle enters the conflict area.

6.5.4 Results

6.6 Conclusion

Chapter 7

Making Teams and Influencing Agents

Efficiently Coordinating Decision Trees for Interpretable Multi-Agent Reinforcement Learning

Domain: Traffic signal control
Challenges: Assurance, coordination

7.1 Introduction

In Chapter 6, I addressed one aspect of *assurance* in deploying AI technologies for transportation: *safety*. I designed an approach for imbuing traffic signal plans based on reinforcement learning (RL) with practical operation constraints. These constraints align the format of the policy with traditional cycle-offset-split plans, making the resulting signal plans are safer and more predictable by stakeholders. In this chapter, I focus on the other aspect of assurance: *interpretability*.

Despite these safety improvements, two challenges still make it difficult for stakeholders to interpret how deep RL policies map states to signalling actions. First, the deep neural network (NN) architectures used for RL policies typically have thousands to millions of parameters. Second, the behaviour of RL policies is difficult to predict and verify over long time horizons. Past literature has found that human stakeholders understand and trust RL policies less than their simpler, rule-based counterparts, even if RL yields superior performance [341]. At the same time, as I noted in Chapter 3, interpretability techniques can help stakeholders trust AI systems more [93, 451].

In traffic signal control (TSC) and other applications where the safety and verifiability of RL policies is critical [119, 156], users may deploy interpretable surrogate policies in place of NN policies. Such surrogate policies should be *performant* — capable of achieving high returns. In multi-agent RL, coordinating the training of surrogates is critical for performance: if multiple surrogates are deployed simultaneously, they cannot assume that they are interacting with performant experts, as their performance may be influenced by suboptimal behaviour from others.

At the same time, surrogate policies should be *computationally efficient* — it should be possible to generate them with minimal environment interactions and runtime. As I noted in Chapter 4,

running a high-fidelity traffic simulator can be computationally intensive. In human-in-the-loop frameworks where users such as traffic engineers provide oversight to correct undesirable policy behaviour [234], the ability to iterate on surrogate policies in a rapid fashion is also critical [411]. The desiderata of performance and computational efficiency exist in tension: more complex models capable of stronger performance and coordination capabilities are also less efficient [252].

Decision trees (DTs) are considered as an attractive model structure for interpretable single-agent RL due to their comprehensibility [338]. They lie at the core of the imitation learning framework VIPER [25], which has been applied to distil deep RL policies into DTs in domains such as TSC [156, 462], autonomous vehicles [320], and robotics [307]. However, generalizing VIPER to the MARL setting is challenging. Prior work [253] has introduced two multi-agent VIPER algorithms, IVIPER and MAVIPER. IVIPER fails to coordinate the training of DTs, thus sacrificing performance; meanwhile, MAVIPER trains DTs in a coordinated but computationally inefficient manner. This makes DT surrogates generated by these algorithms impractical for deployment.

To this end, I introduce HYDRAVIPER, an efficient method to extract coordinated DT policies for cooperative MARL. My method makes three key algorithmic contributions: (1) HYDRAVIPER coordinates agent training by jointly resampling the training dataset for each team of cooperative agents. (2) When interacting with the environment to collect a training dataset, HYDRAVIPER adaptively collects critical trajectories closer to convergence. (3) When interacting with the environment for evaluation, HYDRAVIPER uses a multi-armed bandit-based evaluation strategy to identify promising sets of trained surrogates. Experiments on various environments demonstrate that HYDRAVIPER achieves the goal of balancing performance and computational efficiency. HYDRAVIPER also improves the practical applicability of DT-based interpretable MARL policies: users can exchange training time for performance by altering the algorithm's environment interaction budgets, but its performance remains optimal at different budget levels. Lastly, HYDRAVIPER's efficiency on large environments can be improved while maintaining coordination by dividing the agent set into mutually influential teams.

7.2 Background

In this chapter, I model the multi-agent cooperation problem as a team-based, mixed competitive-cooperative, *partially observable Markov game*, following the formulation fromSection 2.3.3. My algorithm assumes access to value and Q-functions that take the global observations of all agents, not states, as input: $V^{\pi_i}(\mathbf{o})$ and $Q^{\pi_i}(\mathbf{o}, \mathbf{a})$. I define *mean* value functions and state-action value functions for each team: $\bar{V}^{\pi_\ell}(\mathbf{o}) := \frac{1}{|\mathcal{T}_\ell|} \sum_{i \in \mathcal{T}_\ell} V^{\pi_i}(\mathbf{o}), \bar{Q}^{\pi_\ell}(\mathbf{o}, \mathbf{a}) := \frac{1}{|\mathcal{T}_\ell|} \sum_{i \in \mathcal{T}_\ell} Q^{\pi_i}(\mathbf{o}, \mathbf{a})$.

7.2.1 Decision Trees

A *decision tree* (DT) recursively partitions an input space $\mathcal{X}$ through functions $f_j : \mathcal{X} \to \mathbb{R}$ and thresholds τ_j at each internal node j. Each internal node induces a partition of samples, $P_j = x \in \mathcal{X} : f_j(x) \leq \tau_j$. In a DT policy, internal nodes (f_j, τ_j) encode observation-dependent decision criteria, while leaf nodes $l \in \mathcal{L}$ map partitioned observations to actions: $\hat{\pi}_i : \mathcal{O}_i \to \mathcal{A}_i, \forall i \in [N]$.

7.2.2 VIPER

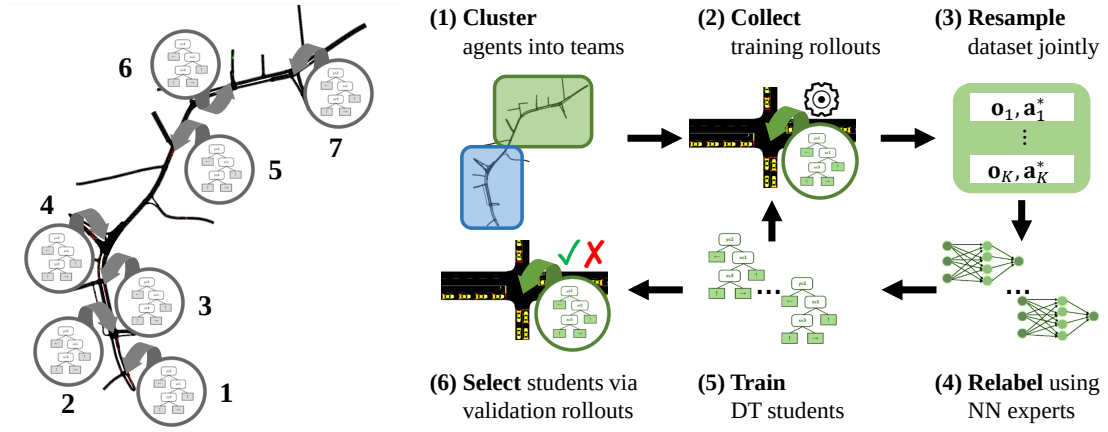


Figure 7.1: (Left) Imitation learning in traffic signal control, where a decision tree must be learned to imitate the RL-based policy of each intersection's signal controller agent (the seven-intersection Ingolstadt corridor TSC environment from Section 7.5.1 is shown, with intersections numbered). (Right) The HYDRAVIPER framework, in which DT students are trained *independently* using a *jointly resampled* dataset of environment trajectories and relabelled by an NN expert.

As I introduced in Section 6.4.2, the goal of *imitation learning* is to train a *student* policy $\hat{\pi}$ based on demonstrations from a *teacher* policy π^* [205], so that the student and teacher achieve similar performance [290]. *VIPER* [25] is an imitation learning framework, adapted from the more general DAGGER [306], that trains DTs as student policies. Given a trained expert (NN) policy π^*, VIPER iteratively generates student (DT) policies $\hat{\pi}^m$. Specifically, in each iteration m, VIPER:

(1) *Collects* K new rollouts $\{\mathbf{o}, \hat{\pi}^{m-1}(\mathbf{o})\}$ using the previous students from iteration $m-1$ (where $\hat{\pi}^0 := \pi^*$)

(2) *Resamples* a dataset $\mathcal{D}$ from all trajectories collected so far, based on upweighting critical states where taking a suboptimal action may be costly in terms of Q-values:

$$p_k \propto V^{\pi^*}(\mathbf{o}_k) - \min_{\mathbf{a}} Q^{\pi^*}(\mathbf{o}_k, \mathbf{a})$$

(3) *Relabels* the dataset with the expert actions $\pi^*(\mathbf{o}_k)$

(4) *Trains* new DT students $\hat{\pi}^m$ on $\{\mathbf{o}_k, \pi^*(\mathbf{o}_k) \mid \mathbf{o}_k \in \mathcal{D}\}$

After M iterations, VIPER (5) *selects* a student through validation on an additional set of rollouts. Ross et al. [306] showed that such a procedure is guaranteed to find a student that is performant on the distribution of states that it induces.

7.3 Related Work

Interpretable Multi-Agent Learning Past methods for interpretable MARL have focused on using feature importances to construct saliency maps [152, 143, 216, 260], logical structures

[168, 390, 157], and domain concepts [432]. Each of these categories of methods has limitations. Feature importances and saliency maps are visually clear, but only highlight aspects of the state space without showing how policies use them. Policies based on logical rules and concepts allow users to align the execution of these policies with domain knowledge, but they require extensive feature engineering. By contrast, I learn simple policy representations grounded directly in the environment feature space.

Decision Trees for Reinforcement Learning Relative to deep NNs, shallow DT policy representations are intrinsically [256] and empirically [338] more comprehensible. One line of work in DT-based RL directly trains DT policies [338, 373, 72, 217] using relaxations amenable to direct optimisation. However, these methods suffer from training instability and performance degradation. Another line of work follows the *VIPER* framework [25], in which a surrogate DT is trained through imitation learning of a performant expert. Although this approach has achieved success in various single-agent settings [320, 307, 156, 462], there exist only two VIPER-based algorithms for the multi-agent setting: *IVIPER* and *MAVIPER* [253]. IVIPER takes a decentralized approach where each agent views other agents as a stationary part of the environment, and therefore independently trains DTs for each agent. This approach is efficient, but the lack of coordination hinders its performance. I show the pseudocode of IVIPER in Algorithm 1. Meanwhile, MAVIPER takes a centralized approach where each agent accounts for the potential impact that its joint actions with others may have, and therefore it jointly trains the DTs by projecting the actions that other agents' DTs would predict. This approach is performant, but the joint DT growth procedure is inefficient. Thus, neither algorithm achieves a balance between performance and computational efficiency.

Algorithm 1 IVIPER

Input: $(\mathcal{S}, \mathcal{A}, P, r_i, o_i)$, π^*, Q^{π^*}, K_{train}, K_{valid}, M
Output: $\hat{\pi}$

Initialize dataset $\mathcal{D} \leftarrow \emptyset$ and policies $\hat{\pi}_i^0 \leftarrow \pi_i^*, \forall i \in N$
for iteration $m \in \{1, \dots, M\}$ **do**
  for each agent $i \in \{1, \dots, N\}$ **do**
    Collect and **relabel** K_{train} training rollouts: $\mathcal{D}_i^m \leftarrow \{(\mathbf{o}, \pi_i^*(o_i)) \sim d^{(\hat{\pi}_i^{m-1}, \pi_{-i}^*)}\}$
    Aggregate dataset $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \mathcal{D}_i^m$
    Set weights for each $(\mathbf{o}_k, \mathbf{a}_k) \in \mathcal{D}_i$: $p_{ik} \leftarrow V^{\pi_i^*}(\mathbf{o}_k) - \min_{a_i} Q^{\pi_i^*}(\mathbf{o}_k, a_i, \pi_{-i}^*(\mathbf{o}_{-ik}))$
    Resample dataset $\mathcal{D}_i' \leftarrow \{(\mathbf{o}_k, \mathbf{a}_k) \sim p_{ik}\}$
    Train DT $\hat{\pi}_i^m \leftarrow$ **TrainDT**$(\mathcal{D}_i')$
    Average return of K_{valid} validation rollouts: $\hat{\mu}_i^m \leftarrow \frac{1}{K_{\text{valid}}} \sum_{k=1}^{K_{\text{valid}}} \bar{r}_{ik}, \bar{r}_{ik} \sim d^{(\hat{\pi}_i^m, \pi_{-i}^*)}$
for each agent $i \in \{1, \dots, N\}$ **do**
  Select best student $\hat{\pi}_i \leftarrow \text{argmax}_m \, \hat{\mu}_i^m$
return best set of agents $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_N)$

7.4 HYDRAVIPER

In this section, I present *HYDRAVIPER* (Algorithm 2), our algorithm for performant and efficient interpretable MARL. As shown in Figure 7.1, HYDRAVIPER builds on the DAGGER and VIPER frameworks by iteratively collecting data from environment rollouts to train DT policies. HYDRAVIPER first (1) *partitions agents* into clusters for scalability (line 4). Next, in each of M iterations, HYDRAVIPER: (2) collects a dataset of rollouts from the environment, using an *adaptive procedure* (lines 6–7); (3) resamples the dataset to prioritise learning the correct actions in critical states, using *team-based Q-values* (lines 9–10); (4) and trains DTs based on these datasets (lines 11–12). After completing all M training iterations, HYDRAVIPER (5) identifies the best-performing student for each agent, using a *multi-armed bandit algorithm*, and returns them as a policy profile (lines 13–14). I now describe each of these algorithm components in detail.

Algorithm 2 HYDRAVIPER

Input: Markov game $(\mathcal{S}, \mathcal{A}, P, R_i, O_i)$, experts π^*, expert Q-functions Q^{π^*}, per-iteration rollout count K_{train}, rollout budgets $(B_{\text{train}}, B_{\text{valid}})$, threshold ϵ, iteration count M, scaling factor c, agent distance function d
Output: Trained students $\hat{\pi}$

1: **Initialise** dataset $\mathcal{D} \leftarrow \emptyset$, policies $\hat{\pi}_i^0 \leftarrow \pi_i^*, \forall i \in N$
2: **Initialise** rollout count $n_{\text{train}} \leftarrow 0$
3: **Initialise** dropped rollout count $K_{\text{drop}} \leftarrow \infty$
▷ Section 7.4.4: Agent Clustering
4: **Cluster** agents $\mathcal{T}_1, \ldots, \mathcal{T}_L \leftarrow$ **Partition**(Γ, π^*, d)
5: **for** $m \in \{1, \ldots, M\}$ **do**
▷ Section 7.4.2: Training Rollouts
6: $\mathcal{D}, n_{\text{train}} \leftarrow$ **TR-A**$(\mathcal{D}, \hat{\pi}^{m-1}, m, K_{\text{train}}, B_{\text{train}}, K_{\text{drop}}, n_{\text{train}})$
7: **Reinitialise** dropped rollout count $K_{\text{drop}} \leftarrow \infty$
8: **for** each team $\mathcal{T}_\ell \in \{\mathcal{T}_1, \ldots, \mathcal{T}_L\}$ **do**
▷ Section 7.4.1: Dataset Resampling
9: $\mathcal{D}'_\ell, K'_{\text{drop}} \leftarrow$ **C-Q**$(\mathcal{D}_\ell, \mathcal{T}_\ell, \pi^*, Q^{\pi^*}, \epsilon)$
10: $K_{\text{drop}} \leftarrow \min(K_{\text{drop}}, K'_{\text{drop}})$
11: **for** each agent $i \in \mathcal{T}_\ell$ **do**
12: $\hat{\pi}_i^m \leftarrow$ **TrainDT**$(\mathcal{D}'_\ell)$
13: **for** each team $\mathcal{T}_\ell \in \{1, \ldots, L\}$ **do**
▷ Section 7.4.3: Validation Rollouts
14: $\hat{\pi}_i, \forall i \in \mathcal{T}_\ell \leftarrow$ **VR-UCB**$(\{\hat{\pi}_\ell^m\}_{m=1}^M, \mathcal{T}_\ell, B_{\text{valid}}, c)$
15: **return** $\hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_N)$

7.4.1 Dataset Resampling: Centralised-Q Weighting

VIPER-based algorithms include a dataset resampling step (Algorithm 2, lines 9–10) so that students can focus their learning on more critical states. At a high level, they construct a training dataset by computing sampling weights over the aggregated dataset of environment rollouts, typically using some notion of value based on the expert Q-functions. Measuring value is straightforward in the single-agent setting, but — as I have mentioned — a key obstacle in multi-agent learning is efficient coordination among agents. To address this challenge, HYDRAVIPER induces coordination in the resampling step using a team-based notion of value (Algorithm 3), but trains DTs independently for each agent.

Algorithm 3 Centralised-Q Resampling (C-Q)

Input: Dataset $\mathcal{D}_\ell$, teams $\mathcal{T}_\ell$, experts π^*, expert Q-functions Q^{π^*}, threshold ϵ
Output: Resampled dataset $\mathcal{D}'_\ell$, dropped rollout count K_{drop}
1: **Set** weights for each $(\mathbf{o}_k, \mathbf{a}_k) \in \mathcal{D}_\ell$: $p_{\ell k} \leftarrow \bar{V}^{\pi^*_\ell}(\mathbf{o}_k) - \min_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k}))$
2: **Update** $K_{\text{drop}} \leftarrow \min_\ell \left\lceil \frac{1}{T} |\{(\mathbf{o}_k, \mathbf{a}_k) \in \mathcal{D}_\ell \mid p_{\ell k} \leq \epsilon\}| \right\rceil$
3: **Resample** dataset: $\mathcal{D}'_\ell \leftarrow \{(\mathbf{o}_k, \mathbf{a}_k) \sim p_{\ell k}\}$
4: **return** $\mathcal{D}'_\ell, K_{\text{drop}}$

Specifically, HYDRAVIPER resamples the dataset for DT construction based on weights $p_{\ell k}$, which compute the relative importance of each sample for each team of agents $\mathcal{T}_\ell$ (Algorithm 3, line 1). Prior work computes this importance based on *individual* Q-functions, meaning that each agent must maintain its own dataset and induce coordination through (typically computationally expensive) joint training procedures. By contrast, I propose an intuitive change: HYDRAVIPER uses the mean of the expert Q-functions within each *team* of coordinated agents, $\bar{Q}^{\pi^*_\ell} := \frac{1}{|\mathcal{T}_\ell|} \sum_{j \in \mathcal{T}_\ell} Q^{\pi^*_j}$, so as to prioritise samples according to their value to the team. Then, I compute the weights as the difference in value between the optimal joint team action and the worst-case joint team action. Intuitively, highly-weighted samples are those where coordinating on joint actions matters for performance. The weights are defined as:

$$\begin{aligned} p_{\ell k} &\propto \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \pi^*(\mathbf{o}_k)) - \min_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k})) \\ &= \bar{V}^{\pi^*_\ell}(\mathbf{o}_k) - \min_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k})). \end{aligned} \tag{7.1}$$

For further gains in sample efficiency, HYDRAVIPER does not compute $p_{\ell k}$ by enumerating joint actions over all agents present in the environment. Instead, it only enumerates possible joint actions $\mathbf{a}_\ell$ over the *team* and uses expert actions $\pi^*_{-\ell}(\mathbf{o}_{-\ell})$ for opponent agents. This novel resampling procedure eliminates the need for per-agent datasets in IVIPER and MAVIPER, allowing agents to prioritise the same critical states without computationally expensive joint training.

HYDRAVIPER uses the jointly sampled dataset to independently train DTs for each agent i (Algorithm 2, lines 8–9). Each DT $\hat{\pi}_i$ uses individual observations o_i to fit π^*_i's actions in the dataset. Modifying the input dataset rather than the training procedure provides HYDRAVIPER with flexibility in the choice of DT learning algorithm. More advanced models such as random forests or mixtures of DTs [381] can be incorporated to improve performance.

7.4.2 Training Rollouts: Adaptive Budget Allocation

Algorithm 4 Adaptive Training Rollouts (TR-A)

Input: Dataset $\mathcal{D}$, students $\hat{\pi}^{m-1}$, iteration m, per-iteration rollout count K_{train}, training rollout budget B_{train}, dropped rollout count K_{drop}, total rollout count n_{train}
Output: Updated dataset $\mathcal{D}$, total rollout count n_{train}
1: **Set** $K^m_{\text{train}} \leftarrow \min(K_{\text{drop}}, K_{\text{train}})\mathbb{1}[n_{\text{train}} \leq B_{\text{train}}]$
2: **Update** $n_{\text{train}} \leftarrow n_{\text{train}} + K^m_{\text{train}}\mathbb{1}[m > 1]$
3: **for** each team $\mathcal{T}_\ell \in \{1, \ldots, L\}$ **do**
4: **Collect and relabel** K^m_{train} rollouts: $\mathcal{D}^m_\ell \leftarrow \{(\mathbf{o}_\ell, \pi^*_\ell(\mathbf{o}_\ell)) \sim d^{(\hat{\pi}^{m-1})}\}$
5: **Aggregate** dataset: $\mathcal{D}_\ell \leftarrow \mathcal{D}_\ell \cup \mathcal{D}^m_\ell$
6: **return** $\mathcal{D}, n_{\text{train}}$

Thus far, I have assumed that HYDRAVIPER has access to a dataset of observation-action pairs for training. To collect this dataset, HYDRAVIPER follows the DAGGER-style iterative procedure of collecting a dataset at each iteration m by rolling out the current student policies $\hat{\pi}^{m-1}$ (Algorithm 2, line 6–7). The next set of students are trained on the aggregate of all collected datasets, therefore building up the set of inputs likely to be encountered by the student policies during execution. However, collecting training rollouts is computationally expensive. Prior work employs an inefficient static allocation strategy which uniformly performs K_{train} rollouts in each iteration. This strategy is problematic because the students are far from convergence early in training, so the distribution of trajectories collected earlier in training potentially diverges from those that converged students would encounter. HYDRAVIPER addresses this challenge through an adaptive rollout strategy that dynamically allocates the training budget at each iteration and prioritises critical states encountered later in training.

Recall that, for each team of cooperative agents $\mathcal{T}_\ell$, HYDRAVIPER follows Equation (7.1) to compute weights $p_{\ell k}$ for resampling the training dataset. I show the following:

Theorem 7.1. *Given a dataset of observation-action pairs for team $\mathcal{T}_\ell$ in iteration m of HYDRAVIPER, $\mathcal{D}_\ell = \{(\mathbf{o}_\ell, \mathbf{a}_\ell)\}$, assume there exists a pair $(\mathbf{o}_{\ell k}, \mathbf{a}_{\ell k})$ that receives the weight $p^{(m)}_{\ell k} = 0$. Then, in iteration $m+1$ of HYDRAVIPER, this pair also receives the weight $p^{(m+1)}_{\ell k} = 0$.*

Proof. If $p^{(m)}_{\ell k} \propto \bar{V}^{\pi^*_\ell}(\mathbf{o}_k) - \min_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k})) = 0$, then by definition

$$\bar{V}^{\pi^*_\ell}(\mathbf{o}_k) := \max_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k})) = \min_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_\ell}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k})), \tag{7.2}$$

i.e. joint team actions $\mathbf{a}_\ell$ have no effect on the value given observation $\mathbf{o}_k$. When HYDRAVIPER resamples the dataset in iteration m (Algorithm 3, line 3), $(\mathbf{o}_{\ell k}, \mathbf{a}_{\ell k})$ will not be part of the resampled dataset $\mathcal{D}'_\ell$. However, the resampled dataset $\mathcal{D}'_\ell$ does not replace the original dataset $\mathcal{D}_\ell$.

In iteration $m+1$, $\mathcal{D}_\ell$ is aggregated with a newly-collected dataset of observation-action pairs $\mathcal{D}^m_\ell$ (Algorithm 4, line 5), and $(\mathbf{o}_{\ell k}, \mathbf{a}_{\ell k})$ continues to be part of this dataset. A new set of weights

$p_\ell^{(m+1)}$ are computed using this expanded dataset (Algorithm 2, line 8). Assume that $p_{\ell k}^{(m+1)} \neq 0$. Without loss of generality, let $p_{\ell k}^{(m+1)} > 0$. Then by definition

$$\bar{V}^{\pi_\ell^*}(\mathbf{o}_k) := \max_{\mathbf{a}_\ell} \bar{Q}^{\pi_\ell^*}(\mathbf{o}_k, \mathbf{a}_\ell, \pi_{-\ell}^*(\mathbf{o}_{-\ell k})) > \min_{\mathbf{a}_\ell} \bar{Q}^{\pi_\ell^*}(\mathbf{o}_k, \mathbf{a}_\ell, \pi_{-\ell}^*(\mathbf{o}_{-\ell k})).$$

None of $\mathbf{o}_k$, $\pi_{-\ell}^*$, $\bar{V}^{\pi_\ell^*}$, or $\bar{Q}^{\pi_\ell^*}$ changed between iteration m and $m+1$, since HYDRAVIPER uses experts and expert value functions to compute the weights. This contradicts Equation (7.2). Thus, if $p_{\ell k}^{(m)} = 0$, then $p_{\ell k}^{(m+1)} = 0$. □

A similar proof holds if the strict equality is replaced by a defined threshold ϵ. As a result, samples $(\mathbf{o}_{\ell k}, \mathbf{a}_{\ell k})$ with $p_{\ell k} \leq \epsilon$ are effectively *removed* from the dataset $\mathcal{D}$. This intuition serves as the motivation behind HYDRAVIPER's adaptive training rollout budget allocation (Algorithm 4): after samples are dropped during the resampling procedure, HYDRAVIPER performs rollouts to replenish the dataset.

Specifically, I treat the first iteration as a warm-up period, in which the experts collect a predefined number of K_{train} rollouts (Algorithm 4, lines 3–5). This leads to an initial dataset of $T \cdot K_{\text{train}}$ observation-action pairs. Each team $\mathcal{T}_\ell$ discards non-critical samples from its dataset (Algorithm 3, line 2), i.e. those where the range in the Q-value is at most a predefined threshold ϵ. With the goal of efficiency in mind, HYDRAVIPER computes the minimum number of such discarded samples across all teams of cooperative agents. This then determines the minimum number of rollouts required to collect at least this many samples in the next iteration. As a result, the expected number of dropped rollouts, and therefore the budget for the next iteration, is:

$$K_{\text{drop}} = \min_\ell \left\lceil \frac{1}{T} |\{(\mathbf{o}_k, \mathbf{a}_k) \in \mathcal{D}_\ell \mid p_{\ell k} \leq \epsilon\}| \right\rceil.$$

Over the remaining $M-1$ iterations, HYDRAVIPER continues collecting rollouts using students until it exhausts its total budget of B_{train} training rollouts (Algorithm 4, line 1). Choosing different rollout budgets allows performance and efficiency to be traded off. A higher budget is likely to lead to superior performance, as more rollouts will be collected from students closer to convergence before the budget is exhausted, but it also requires more computation time.

7.4.3 Validation Rollouts: UCB Policy Selection

Following M iterations, HYDRAVIPER produces M joint policy profiles for each team. It then needs to select the best-performing policy profile (Algorithm 2, lines 13–14). HYDRAVIPER iterates through the policy profiles to estimate the team performance of each one using a set of validation rollouts. The performance metric it uses is the undiscounted mean return of the team, $\bar{R}_\ell^m = \frac{1}{T} \sum_{t=0}^{T} R_\ell(s_t, \hat{\pi}_\ell^m(\mathbf{o}_t))$.

As is the case for training, collecting validation rollouts is computationally intensive, so these rollouts also need to be efficiently allocated. However, the problem setting differs here. The goal is not to collect a *diverse* set of training rollouts, but rather to identify the *most performant* policy profiles using as few rollouts as possible. The mean return of each policy profile is unknown *a*

Algorithm 5 UCB Validation Rollouts (VR-UCB)

Input: Policies $\{\hat{\pi}_\ell^m\}_{m=1}^M$, team $\mathcal{T}_\ell$, validation rollout budget B_{valid}, scaling factor c
Output: Selected policies $\hat{\pi}_i, \forall i \in \mathcal{T}_\ell$

Initialise $n_m \leftarrow 0$ for all $m \in \{1, \dots, M\}$
Initialise $n_{\min} \leftarrow \lceil 2 \ln B_{\text{valid}} \rceil$
Initialise return estimates: $\mu_\ell^m \leftarrow \frac{1}{C_{min}} \sum_{k=1}^{C_{min}} \bar{R}_{\ell k}, \bar{R}_{\ell k} \sim d^{(\hat{\pi}_\ell^m, \pi_{-\ell}^*)}$
for rollout $k \in \{1, \dots, (B_{valid} - m n_{\min})\}$ **do**
    Set $m^* \leftarrow \text{argmax}_m \hat{\mu}_\ell^m + \sqrt{\frac{c \ln B_{\text{valid}}}{n_m}}$
    Collect mean return: $\bar{R}_{\ell k} \sim d^{(\hat{\pi}_\ell^{m^*}, \pi_{-\ell}^*)}$
    Update rollout count: $n_{m^*} \leftarrow n_{m^*} + 1$
    Update running average of mean return: $\hat{\mu}_\ell^{m^*} \leftarrow \frac{n_{m^*}-1}{n_{m^*}} \hat{\mu}_\ell^{m^*} + \frac{1}{n_{m^*}} \bar{R}_{\ell k}$
return $\hat{\pi}_\ell^{m^*}, \forall i \in \mathcal{T}_\ell$

priori; it must be estimated by selecting policy profiles and performing rollouts with noisy returns. Again, a fixed allocation strategy of K_{valid} environment rollouts for each policy profile is wasteful. The rollouts assigned to clearly badly-performing policy profiles could be reallocated to reduce the variance in the estimated returns of promising policy profiles. This motivation aligns with that of multi-armed bandit (MAB) problems.

Given a limited budget of B_{valid} rollouts, I represent the task of selecting the best-performing policy profile as a MAB problem. For each team $\mathcal{T}_\ell$, the policy profile $\hat{\pi}_\ell^m$ from each iteration m is an arm, and its return is a random variable $\bar{R}_\ell^m$ with unknown mean μ_ℓ^m. Each rollout samples from one such random variable, which captures the distribution of returns from environment and policy randomness. The objective is to identify the best arm $m_\ell^* = \text{argmax}_m \mu_\ell^m$ in as few rollouts as possible, i.e. to minimise the regret with respect to the policy that selects m_ℓ^* for every rollout.

In this work, I use a modification of the UCB1 algorithm [12]. This allows us to achieve logarithmic regret given a readily satisfiable assumption: that the returns $\bar{R}_\ell^m$ of the arms are bounded. Although UCB assumes that the arms are bounded in $[0, 1]$, it can be modified in a manner equivalent to rescaling the rewards to remain in $[0, 1]$. I rely on the general form of the Chernoff-Hoeffding bound:

Theorem 7.2. *(Theorem 2 of Hoeffding [146]) For independent random variables $X_1, \dots, X_n$ with mean μ, $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$, and $a_i \leq X_i \leq b_i, \forall i \in \{1, \dots, n\}$, then for $\alpha > 0$*

$$\Pr(\bar{X} \geq \mu + \alpha) \leq e^{-\frac{2n^2\alpha^2}{\sum_{i=1}^n (b_i - a_i)^2}}.$$

Corollary 7.1. *Assume that $\bar{R}_\ell^m$ is bounded by $[a, b]$ with $\Delta = b - a$ for all i, m. For $c = 2\Delta^2$, Theorem 7.2 shows that, for the empirical mean $\hat{\mu}_\ell^m$, (notation simplified for clarity)*

$$\Pr\left(\hat{\mu}_\ell^m \geq \mu_\ell^m + \sqrt{\frac{c \ln B}{n_m}}\right) \leq e^{-\frac{2c \ln B}{\Delta^2}} = B^{-4}.$$

This is the same bound demonstrated for UCB1, and the same holds for the lower confidence bound $\mu_\ell^m - \sqrt{\frac{c \ln B}{n_m}}$. Overall, this choice of c leads to the same $O(\log B)$ regret bound as UCB1. However, HYDRAVIPER can also be extended to use other MAB algorithms. If the mean returns of each policy profile are assumed to be normally distributed, the UCB1-NORMAL algorithm [12] could be used; it also achieves logarithmic regret. This algorithm effectively chooses c to be proportional to the arms' sample variance; the greater the variance, the wider the confidence bound. It is also possible to obtain an offline (but biased) estimate of the sample variance by performing expert rollouts before running UCB.

Given a total budget of B_{valid} validation rollouts, HYDRAVIPER performs them as follows. For each policy profile, it first performs $n_{\min} = \lceil 2 \ln B_{\text{valid}} \rceil$ rollouts to generate initial estimates of the mean returns (Algorithm 5, lines 2–3). To allocate the remainder of the budget (lines 4–8), HYDRAVIPER follows UCB1 to select the policy profile index for the kth validation rollout as

$$m^*_{\ell k} = \operatorname*{argmax}_m \left(\hat{\mu}_\ell^m(k) + \sqrt{\frac{c \ln B_{\text{valid}}}{n_m(k)}} \right),$$

where $n_m(k) = \sum_{k'=1}^{k} \mathbb{1}[m^*_{\ell k'} = m]$ is the number of rollouts that have used policy profile m thus far, $\hat{\mu}_\ell^m(k) = \frac{\sum_{k'=1}^{k} \bar{R}_{\ell k'} \mathbb{1}[m^*_{\ell k'} = m]}{n_m(k)}$ is the empirical mean of the returns $\bar{R}_{\ell k}$ from policy profile m, B_{valid} is the total budget of rollouts, and c is a scaling constant for the confidence bound (see Section 7.5.4). HYDRAVIPER maintains a running average for the mean return of each policy profile, which it updates using the mean return $\bar{R}_{\ell k}$ of each rollout (line 8).

7.4.4 Agent Clustering: Scaling Up HYDRAVIPER

Algorithm 6 Agent Graph Clustering (Partition)

Input: Markov game $(\mathcal{S}, \mathcal{A}, P, R_i, O_i)$, experts π^*,
agent distance function d
Output: Agent teams $\mathcal{T}_1, \ldots, \mathcal{T}_L$
1: **Construct** graph $G = (V = \{1, \ldots, N\}, E, w = 0)$
2: **for** each agent $i \in \{1, \ldots, N\}$ **do**
3: **for** each agent $j \in \{1, \ldots, N\}$ **do**
4: **Assign** edge weight $w_{ij} \leftarrow \frac{1}{d(i,j)}$
5: **Partition** graph $\mathcal{T}_1, \ldots, \mathcal{T}_L \leftarrow$ **METIS**(G, L)
6: **return** $\mathcal{T}_1, \ldots, \mathcal{T}_L$

When resampling the dataset, HYDRAVIPER calculates sample weights using Equation (7.1). This computation requires enumerating joint actions $\mathbf{a}_\ell$ for each team $\mathcal{T}_\ell$, in order to find the worst-case joint action that minimises the team-averaged Q-function, $\min_{\mathbf{a}_\ell} \bar{Q}^{\pi^*_i}(\mathbf{o}_k, \mathbf{a}_\ell, \pi^*_{-\ell}(\mathbf{o}_{-\ell k}))$. The complexity of this step scales with the size of the joint action space, and thus exponentially with the size of the team. Some mixed competitive-cooperative environments (see Section 7.5.1) have

inherent team structure that can reduce this complexity. In cooperative environments such as TSC, HYDRAVIPER clusters the agent set into teams to improve training efficiency (Algorithm 4).

The goal is to find a clustering of the agent set into teams $\mathcal{T}_1 \dots \mathcal{T}_L$, so that HYDRAVIPER-trained DT students have *performance* similar to training on the full agent set, but improved *scalability* in that the number of actions to enumerate per team is much smaller than the full agent set: $\prod_{i \in \mathcal{T}_\ell} \mathcal{A}_i \ll \prod_{i \in \{1,\dots,N\}} \mathcal{A}_i$. I leverage the intuition that agents distant from each other (in terms of environmental distance, trajectory similarity, or other metrics) are unlikely to be influential on each other in most environments.

Suppose that there is a function $d(i, j)$ that computes this distance between a pair of agents. In my clustering procedure (Algorithm 6), I first construct a complete graph $G = (V, E) = K_N$ where the nodes represent agents, and the weight between node i and node j, w_{ij}, is inversely proportional to $d(i, j)$ (lines 1–4). Then, I perform *graph partitioning* to divide G into L contiguous, connected node clusters of approximately equal size (line 5), such that the sum of the weights of inter-cluster edges is minimised. I use the hierarchical METIS algorithm [167] to accomplish this. Note that I solve a graph partitioning problem instead of a min-cut problem to prevent the clusters from being imbalanced. Otherwise, in the worst case, the largest cluster could have size $O(N)$, thus yielding minimal gains in scalability.

How can the distance metric d be defined? For an environment that has inherent team structure (such as the physical deception environment in Section 7.5.1), I define the graph G to consist of a complete subgraph for each team:

$$d_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ share a team} \\ \infty & \text{otherwise} \end{cases}$$

For traffic signal control environments, I note that the road network inherently forms a graph G_{env}, which can be partitioned to obtain sets of spatially proximal agents that correspond to neighbouring intersections. In this case, I simply set $G = G_{env}$, or equivalently

$$d_{ij} = \begin{cases} 1 & e_{ij} \in G_{env} \\ \infty & e_{ij} \notin G_{env} \end{cases}$$

7.5 Experiments

Now, I demonstrate the utility of HYDRAVIPER for interpretable MARL using experiments on various benchmark environments. In doing so, I perform a functionally grounded evaluation of interpretability [91], where I assess the quality of the generated DTs in terms of *performance* and *computational efficiency*. As the DTs would be used directly in lieu of NN-based policies in deployment, I consider these to be good proxy metrics of their practical applicability. More specifically, I address the following research questions:

RESEARCH QUESTION 7.1. *Is HYDRAVIPER both performant and efficient (in terms of environment interactions and runtime)?*

RESEARCH QUESTION 7.2. *Does HYDRAVIPER maintain performance optimality as the environment interaction budget decreases?*

RESEARCH QUESTION 7.3. *Can HYDRAVIPER maintain performance optimality while scalability is improved through agent clustering?*

7.5.1 Environments

I evaluate HYDRAVIPER on four environments: two environments in the *multi-agent particle world* (MPE) benchmark [219], and two *traffic signal control* (TSC) environments in the RESCO benchmark [14]. In MPE environments, agents must navigate in a 2D space to accomplish a coordinated objective, making these environments ideal for assessing coordination capabilities.

Cooperative navigation (CN) In this environment, a team of three agents must coordinate to split up and cover three different targets while avoiding collisions with each other.

Physical deception (PD) In this environment, a team of two defender agents must cooperate to protect two targets from an adversary agent. One of the two targets is the "goal" for the adversary; this is not known to the adversary, which can only observe the positions of the targets and defenders. I train the two defender agents against an NN adversary.

In TSC environments, each agent controls a single intersection by selecting different signal phases; each phase permits vehicles from a subset of lanes to pass through the intersection. Both environments are based on real-world road corridors reproduced in the traffic simulator SUMO [10]. To interface with the simulator, I use the OpenAI Gym-style wrapper `sumo-rl` [6]. I focus on imitating experts for all agents as a team.

Cologne corridor (CC) [377] This environment is the same three-agent environment from the experimental evaluation in Chapter 6.

Ingolstadt corridor (IC) [218] This environment is the same seven-agent environment from the experimental evaluation in Chapter 4.

7.5.2 Baselines and Setup

I compare HYDRAVIPER with IVIPER and MAVIPER, which represent the state of the art in interpretable multi-agent RL with DT surrogate policies. I additionally compare with *expert* policies — MADDPG [219] for MPE and MPLight [54] for TSC — and an additional baseline, *imitation DT*. Imitation DT does not use students to collect rollouts, nor does it perform dataset resampling; it collects the same number of training rollouts as the other algorithms and trains DTs on the collected dataset. As imitation DT performs worse than the other algorithms by a wide margin, I only show its performance in Table 7.2.

For MPE environments, I use a horizon of 25 timesteps per episode, and I trained MADDPG for 60 000 episodes as the expert for the DT students to imitate. For TSC environments, I use a horizon of 125 timesteps per episode (each timestep represents 20 seconds of simulation time), and I trained MPLight for 500 episodes as the expert. All imitation learning algorithms were run for 100 iterations to produce DTs with a maximum depth of 4. IVIPER and MAVIPER ran $K_{\text{train}} = K_{\text{valid}} = 50$ training and validation rollouts per iteration for MPE (including for the initial iteration where rollouts are collected by the experts), and 10 rollouts per iteration for TSC. Imitation DT ran the same numbers of training rollouts. I set these to equalise the number of environment interactions per iteration.

I repeated all experiments 10 times with different random seeds, and I report the mean and 95% confidence interval of the reward over 10 rollouts performed with the final student policy profiles generated from these runs. Most experiments were run in parallel on a server with 56 2.75GHz AMD EPYC 7453 processors and 252 GiB of RAM. For these experiments, I report the *number of rollouts collected*, not *runtime*s, as the time per rollout is roughly constant. However, I also report runtimes for the execution of IVIPER, MAVIPER, and HYDRAVIPER on all four environments. For these experiments, I use the `kernprof` profiler (v4.1.3) to run them in sequence, with no other concurrent processes running except system routines. These experiments were run on another server with 8 4.2GHz Intel i7-7700K processors and 62 GiB of RAM.

7.5.3 Results

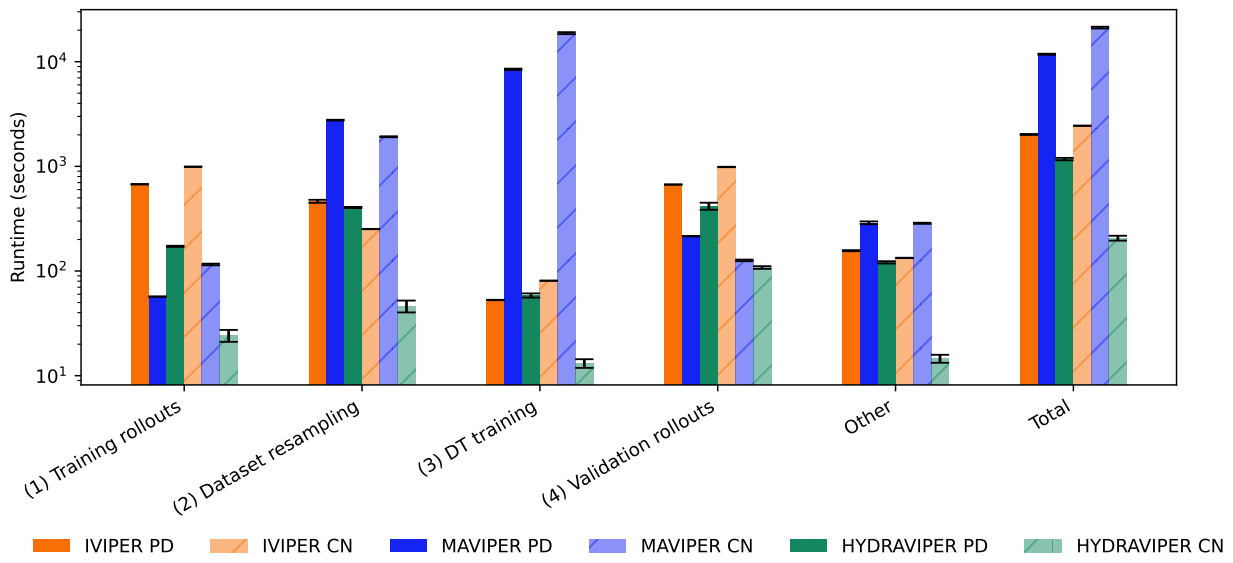


Figure 7.2: Runtime decomposition for IVIPER, MAVIPER, and HYDRAVIPER (with full environment interaction budget) on the multi-agent particle world environments, physical deception and cooperative navigation. Error bars show 95% confidence intervals based on 10 random seeds.

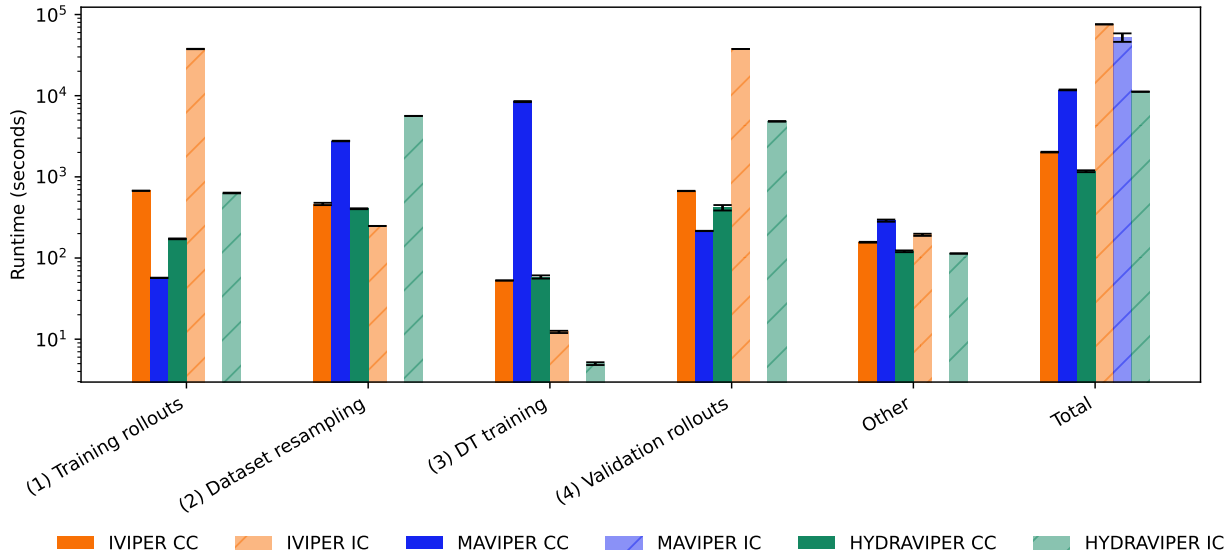


Figure 7.3: Runtime decomposition for IVIPER, MAVIPER, and HYDRAVIPER (with full environment interaction budget) on the traffic signal control environments, Cologne corridor and Ingolstadt corridor. Error bars show 95% confidence intervals based on 10 random seeds.

Research Question 7.1 HYDRAVIPER achieves strong, coordinated performance in a computationally efficient manner. First, I assess HYDRAVIPER's performance as I vary it between two environment interaction budget levels, high (5 000 training/5 000 validation rollouts for MPE, 1 000 training/1 000 validation rollouts for TSC) and low (500 training/1 500 validation rollouts for MPE, 100 training/100 validation rollouts for TSC). As shown in Table 7.1, HYDRAVIPER students perform better than or comparable to students trained by the most performant DT baseline (MAVIPER for MPE, IVIPER for TSC) on all environments at both budget levels. HYDRAVIPER's performance is also better than or comparable to the NN experts for all environments except cooperative navigation, in which all DT-based algorithms cannot achieve expert-level performance. In physical deception, although neither MAVIPER nor HYDRAVIPER substantially outperforms IVIPER given the considerable stochasticity in the environment, HYDRAVIPER achieves a level of performance much closer to MAVIPER, while its training time is an order of magnitude less than MAVIPER.

In the TSC environments, HYDRAVIPER is the best-performing algorithm at both high and low interaction budget levels. Notably, HYDRAVIPER at the high budget level substantially outperforms the expert on the Ingolstadt corridor. By contrast, MAVIPER fails to coordinate the intersection agents and is overall the worst-performing algorithm. HYDRAVIPER more than halves the runtime from both IVIPER and MAVIPER on both TSC environments.

Research Question 7.2 As the environment interaction budget decreases, HYDRAVIPER still outperforms baselines. Now, I investigate the ability of HYDRAVIPER to adapt to increasing budget constraints for environment interaction, as would be imposed by users who wish to

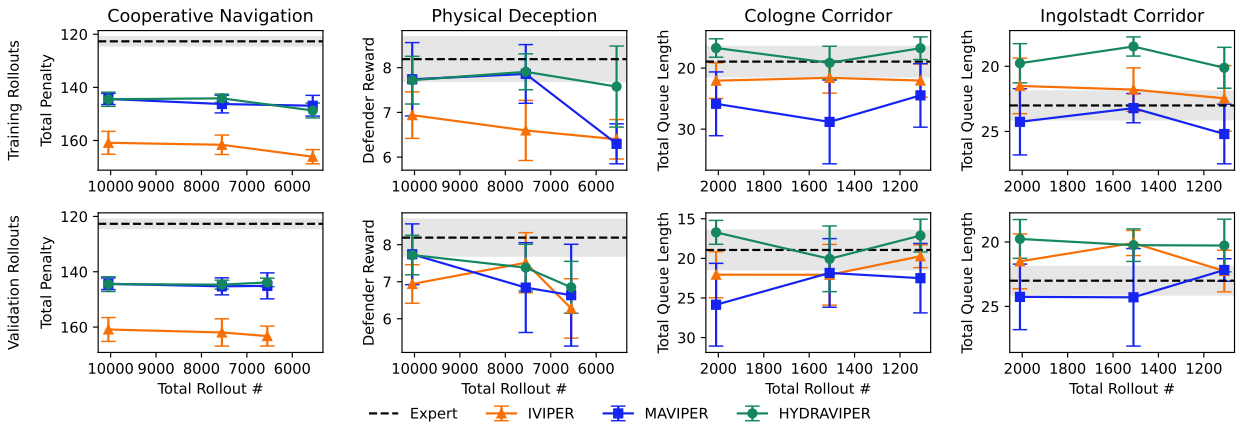


Figure 7.4: Performance of HYDRAVIPER and baselines as the number of rollouts decreases. Top shows decreasing training rollouts; bottom shows decreasing validation rollouts. HYDRAVIPER's performance stays consistent as the number of rollouts decreases. For physical deception, higher rewards are better; for all other environments, lower rewards are better. Bars show 95% confidence intervals based on 10 randomly-seeded runs. Full results are shown in Table 7.2.

quickly iterate on DT policy training. As shown in Figure 7.4, HYDRAVIPER's performance in all four environments does not substantially change as the training and validation rollout budgets are reduced individually. Furthermore, in all four environments, HYDRAVIPER achieves performance on par with or better than MAVIPER at all budget levels. Therefore, HYDRAVIPER is able to maintain a Pareto frontier in the tradeoff between performance and computational efficiency.

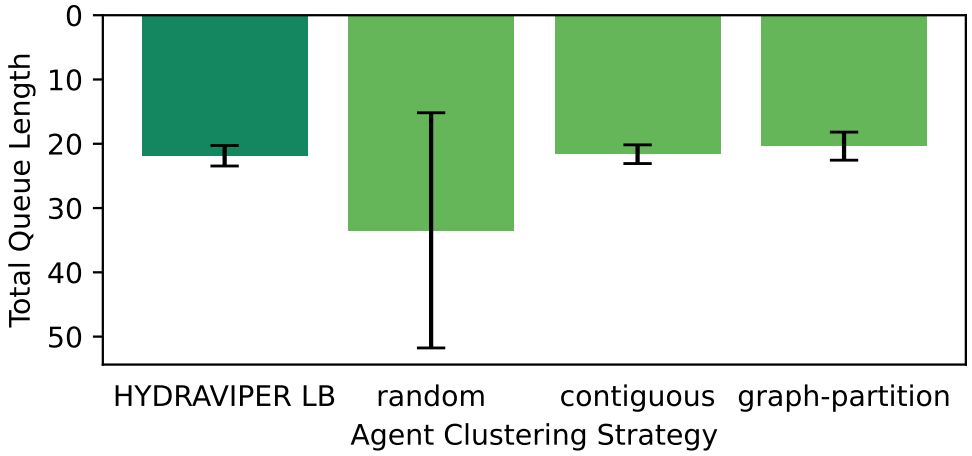
In cooperative navigation, the performance of both HYDRAVIPER and MAVIPER remain similar as the training and validation budgets are reduced individually. However, when both budgets are reduced simultaneously (shown in Figure 7.10), the performance of HYDRAVIPER but not MAVIPER remains essentially unchanged. In physical deception, HYDRAVIPER still performs well even as its training budget is reduced by a factor of 10, whereas MAVIPER performs substantially worse. Furthermore, the 95% confidence intervals of HYDRAVIPER's rewards are smaller than MAVIPER at all validation budget levels. Thus, HYDRAVIPER is able to identify performant policy profiles more consistently than MAVIPER.

In the Cologne corridor, HYDRAVIPER's performance consistently remains within the expert's 95% confidence interval at all environment interaction budget levels, whereas the same is not true of MAVIPER. Meanwhile, the performance of HYDRAVIPER on the Ingolstadt corridor substantially exceeds the expert at all budget levels, whereas MAVIPER and IVIPER (except for the 500 validation rollout setting) remain in the expert's 95% confidence interval.

Research Question 7.3 Even when the agent set is decomposed through clustering, HYDRAVIPER maintains its performance. Finally, I evaluate the effect of agent clustering on the performance of HYDRAVIPER. For physical deception, HYDRAVIPER by default runs with the adversary and the two defenders on opposing teams, and Table 7.1 and Figure 7.4 show that it achieves good performance. For the Ingolstadt corridor environment in the low-budget setting

(100 training/100 validation rollouts), I evaluate the effect of directly clustering the agent set into two teams based on the road network graph G_{env}. As shown in Figure 7.5, the graph partitioning method introduced in Section 7.4.4 allows HYDRAVIPER to retain its performance even when the size of the agent set is approximately halved for each team. This reduces the size of the joint action set for the largest team from 1 944 to 72, and the total runtime of HYDRAVIPER by 20%. This is a notable improvement given the relatively small size of the dataset in the low-budget setting.

My graph partitioning method also outperforms two baselines. First, its mean reward and reward variance are both much smaller than a `random` agent clustering method, in which agents are randomly assigned to one of two teams. Second, graph partitioning delivers a small performance improvement over the `contiguous` clustering method, which randomises between all possible cuts of the road network graph into two connected components. The strong performance of this baseline suggests that incorporating environmental structure is important to a high-quality clustering. Unlike the two baselines, however, my graph partitioning method always yields a clustering that is evenly distributed in size. The resulting clustering also corresponds to the cut under the `contiguous` method with the best performance.



Clustering Method	**Runtime (s)**	**Worst-Performing Clustering**
HYDRAVIPER LB	6 490.0 ± 98.0 s	[[1,2,3,4,5,6,7]]
+ random	5 130.7 ± 104.2 s	[[1,2,6],[3,4,5,7]]
+ contiguous	5 135.6 ± 99.0 s	[[1,2,3,4,5],[6,7]]
+ graph-partition	5 181.8 ± 149.3 s	[[1,2,3],[4,5,6,7]]

Figure 7.5: (Top) Performance of HYDRAVIPER on the Ingolstadt corridor (IC) under different agent clustering methods at the low budget level. (Bottom) Runtimes and resulting clusterings (worst-performing clustering across 10 different random seeds shown) of HYDRAVIPER under these methods. For the clusterings, the intersection agent numbers follow those shown in Figure 7.1. Note that `graph-partition` achieves the best performance and runtime.

Environment		Expert	IVIPER	MAVIPER
Cooperative Navigation	*Total Penalty*	122.67 ± 1.67	160.87 ± 4.31	144.35 ± 2.12
	Runtime (s)	N/A	$2\,444.6 \pm 9.1$	$21\,188.7 \pm 408.6$
Physical Deception	*Defender Reward*	8.19 ± 0.50	6.94 ± 0.52	7.74 ± 0.82
	Runtime (s)	N/A	$2\,017.2 \pm 21.3$	$11\,782.4 \pm 137.8$
Cologne Corridor	*Queue Length*	18.94 ± 2.49	22.06 ± 2.91	25.85 ± 5.22
	Runtime (s)	N/A	$33\,841.4 \pm 441.3$	$37\,503.8 \pm 834.8$
Ingolstadt Corridor	*Queue Length*	23.01 ± 1.10	21.51 ± 2.13	24.26 ± 2.54
	Runtime (s)	N/A	$75\,709.6 \pm 441.3$	$52\,316.9 \pm 6\,280.2$

Environment		HYDRAVIPER	HYDRAVIPER (LB)
Cooperative Navigation	*Total Penalty*	144.48 ± 2.67	144.84 ± 2.12
	Runtime (s)	206.2 ± 11.1	180.5 ± 9.3
Physical Deception	*Defender Reward*	7.72 ± 0.53	7.12 ± 0.84
	Runtime (s)	1173.5 ± 21.6	388.4 ± 5.6
Cologne Corridor	*Queue Length*	16.72 ± 1.51	18.77 ± 3.69
	Runtime (s)	$13\,651.6 \pm 254.2$	1865.4 ± 26.1
Ingolstadt Corridor	*Queue Length*	19.77 ± 1.51	21.87 ± 1.59
	Runtime (s)	$11\,203.9 \pm 67.2$	$6\,490.0 \pm 98.0$

Table 7.1: Performance and runtimes (means and 95% confidence intervals) for HYDRAVIPER and baselines. All algorithms are given the same environment interaction budget, except for low-budget (LB) HYDRAVIPER (which uses 20% of the rollouts for MPE, 10% of the rollouts for TSC). HYDRAVIPER achieves or exceeds the performance of MAVIPER using a fraction of the runtime, and still performs well in the low-budget setting. For physical deception, higher rewards are better; for all other environments, lower rewards are better. Figure 7.2 and Figure 7.3 show runtimes for individual algorithm steps.

Environment	Training	Validation	Expert	Imitation DT	IVIPER	MAVIPER	HYDRAVIPER
Cooperative Navigation	5000	5000	122.67 ± 1.67	221.19 ± 8.58	160.87 ± 4.31	144.35 ± 2.12	144.48 ± 2.67
	2500	5000		211.84 ± 6.32	161.62 ± 3.66	146.28 ± 3.34	144.13 ± 1.59
	500	5000		218.22 ± 5.90	166.11 ± 2.67	146.91 ± 3.90	148.71 ± 2.80
	5000	2500			161.94 ± 4.93	145.27 ± 3.07	144.66 ± 1.62
	5000	1500			163.25 ± 3.60	145.10 ± 4.69	143.86 ± 1.54
	500	1500			168.31 ± 5.52	153.66 ± 4.07	144.84 ± 2.12
Physical Deception	5000	5000	8.19 ± 0.50	6.27 ± 0.43	6.94 ± 0.52	7.74 ± 0.82	7.72 ± 0.53
	2500	5000		5.73 ± 0.31	6.60 ± 0.67	7.84 ± 0.66	7.91 ± 0.40
	500	5000		5.32 ± 0.61	6.40 ± 0.44	6.30 ± 0.45	7.58 ± 0.91
	5000	2500			7.51 ± 0.81	6.84 ± 1.21	7.38 ± 0.63
	5000	1500			6.28 ± 0.80	6.64 ± 1.37	6.85 ± 0.70
	500	1500			6.03 ± 0.70	7.36 ± 0.99	7.12 ± 0.84
Cologne Corridor	1000	1000	18.94 ± 2.49	137.67 ± 0.64	22.06 ± 2.91	25.85 ± 5.22	16.72 ± 1.51
	500	1000		211.84 ± 8.00	21.60 ± 2.50	28.82 ± 6.86	19.13 ± 2.70
	100	1000		218.22 ± 10.29	22.05 ± 2.72	24.47 ± 5.21	16.75 ± 1.85
	1000	500			22.07 ± 3.83	21.84 ± 4.33	20.06 ± 4.15
	1000	100			19.73 ± 1.43	22.50 ± 4.38	17.12 ± 2.07
	100	100			23.40 ± 3.53	28.91 ± 7.27	18.77 ± 3.69
Ingolstadt Corridor	1000	1000	23.01 ± 1.10	169.43 ± 5.26	21.51 ± 2.91	24.26 ± 2.54	19.77 ± 1.51
	500	1000		170.55 ± 3.41	21.79 ± 1.67	23.21 ± 1.12	18.48 ± 0.74
	100	1000		166.99 ± 3.67	22.46 ± 2.52	25.20 ± 2.29	20.11 ± 1.57
	1000	500			20.08 ± 0.97	24.30 ± 3.79	20.25 ± 1.26
	1000	100			22.26 ± 1.62	22.18 ± 0.87	20.28 ± 2.04
	100	100			27.23 ± 5.31	23.31 ± 1.44	21.87 ± 1.59

Table 7.2: Performance (means and 95% confidence intervals) for IVIPER, MAVIPER, and HYDRAVIPER at different training and validation budget levels. For physical deception, higher rewards are better; for all other environments, lower rewards are better.

7.5.4 Hyperparameter Sensitivity

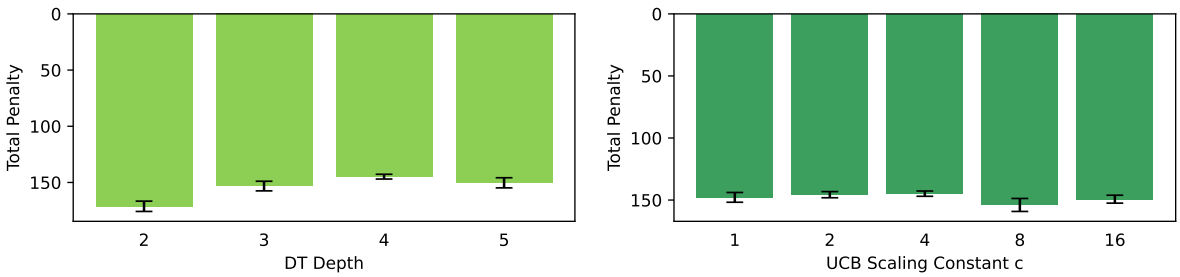


Figure 7.6: Sensitivity of HYDRAVIPER to two hyperparameters, DT depth and the UCB scaling constant c, on the cooperative navigation (CN) environment. Lower rewards are better. The default value is 4 for both hyperparameters.

To understand the effects of HYDRAVIPER's hyperparameters on its performance, we conduct experiments to vary the depth of the DT students, and the scaling constant c for UCB policy selection (Section 7.4.3), on the cooperative navigation environment. We choose this environment due to its relatively low level of randomness. For these experiments, we use HYDRAVIPER at the low budget level (500 training/1 500 validation rollouts) as the baseline algorithm, and fix all hyperparameters other than those of interest.

By default, we use a DT depth of 4; our results show that DTs of this depth provide a good tradeoff between expressiveness and computational efficiency. As shown on the left of Figure 7.6, depth-4 DTs outperform depth-2 and depth-3 DTs on cooperative navigation. This is an intuitive result; the optimal agent policy for this environment cannot be represented with such shallow DTs, as they must condition on the positions of the other agents and the landmarks. However, we find that depth-4 DTs also marginally outperform depth-5 DTs. This same pattern exists in all of the environments that we use for evaluation. We hypothesise that the amount of data collected by HYDRAVIPER at the low budget level is insufficient to coordinate between depth-5 DTs.

Our default value for c is also 4. As we outlined in Corollary 7.1 in Section 7.4.3, the value that theoretically allows logarithmic regret to be achieved is based on the range of mean returns in the environment. Since the agents navigate in a 2×2 square environment, the maximum distance of an agent to a target is $2\sqrt{2}$. The reward in this environment is the negation of the minimum agent distance to each landmark, plus a penalty of -1 for each agent that the ego agent collides with. Therefore, the maximum possible penalty is $\Delta = 3 \cdot 2\sqrt{2} + 2$, which requires $c \approx 219.88$ to achieve the guarantee of Corollary 7.1. However, as shown on the right of Figure 7.6, $c = 4$ empirically performs best; $c = 8$ and $c = 16$ are already excessively conservative given the low randomness in the environment. Since the agents are already trained, we hypothesise that the potential range of returns is less useful in practice for finding a good policy profile than the typical range of returns. This can be approximated by expert rollouts, as we described in Section 7.4.3.

Sensitivity results for the three other environments are also shown below. The best DT depths for these environments are all 4, as with cooperative navigation, while the best tested values of c for physical deception and the Cologne corridor are respectively 2 and 16.

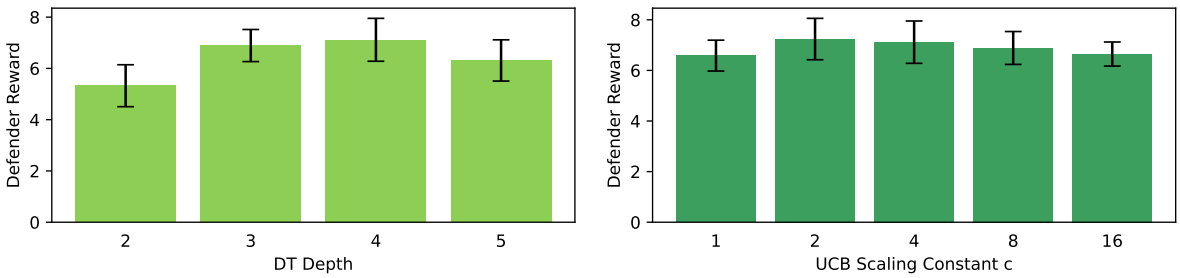


Figure 7.7: Sensitivity of HYDRAVIPER to two hyperparameters, DT depth and the UCB scaling constant c, on the physical deception (PD) environment. Higher rewards are better.

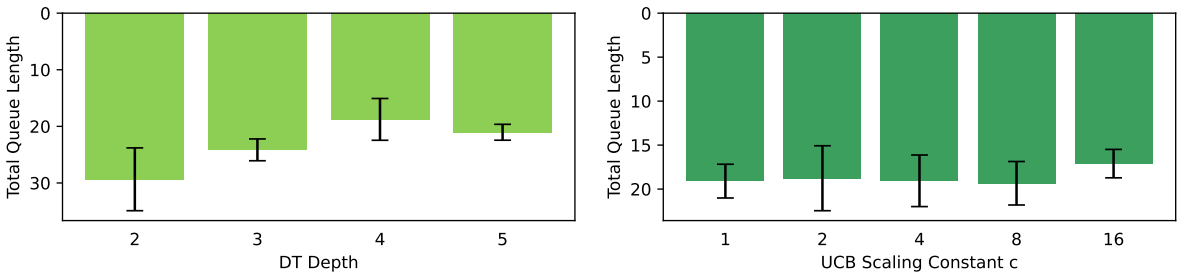


Figure 7.8: Sensitivity of HYDRAVIPER to two hyperparameters, DT depth and the UCB scaling constant c, on the cooperative navigation (CN) environment. Lower rewards are better. The default value is 4 for both hyperparameters.

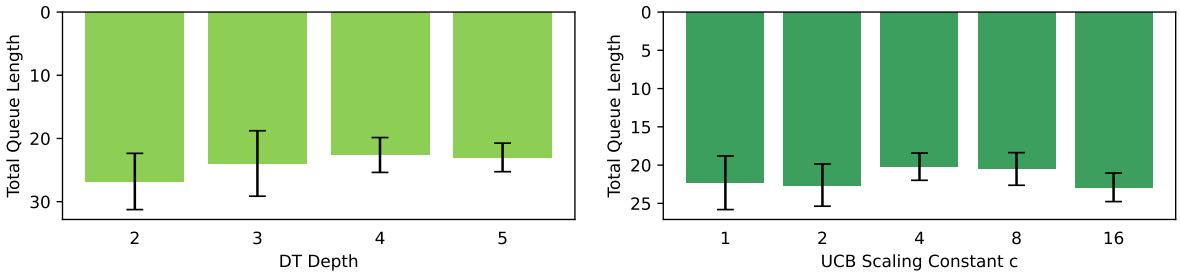


Figure 7.9: Sensitivity of HYDRAVIPER to two hyperparameters, DT depth and the UCB scaling constant c, on the Ingolstadt corridor (IC) environment. Lower rewards are better.

7.5.5 Ablation

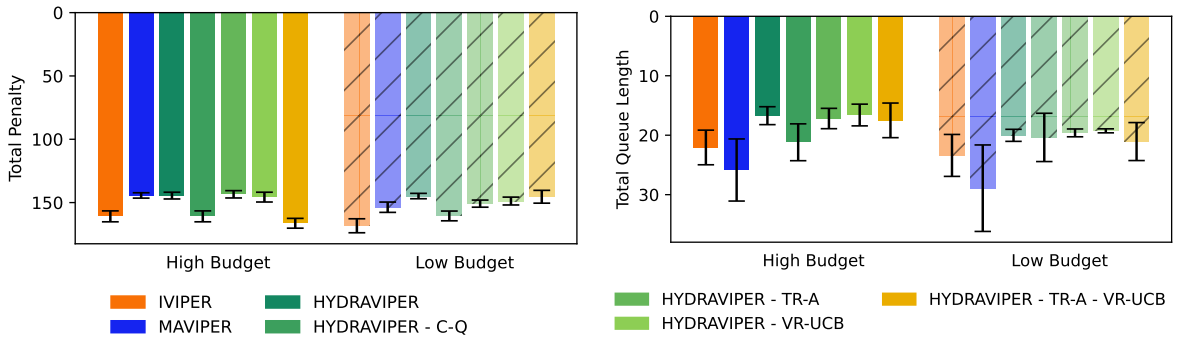


Figure 7.10: Ablation of HYDRAVIPER on the cooperative navigation (CN, left) and Cologne corridor (CC, right) environments. Lower rewards are better.

Lastly, to understand which components of HYDRAVIPER are responsible for its success, we conduct an ablation study for HYDRAVIPER at two budget levels in all four environments. For the high budget level, we use 5 000 training/5 000 validation rollouts for MPE environments, and 1 000 training/1 000 validation rollouts for TSC environments; for the low budget level, we use 500 training/1 500 validation rollouts for MPE environments, and 100 training/100 validation rollouts for TSC environments. We compare HYDRAVIPER's centralised-Q resampling with IVIPER's independent resampling (HYDRAVIPER - CQ). Also, we study the impact of removing adaptive training budget allocation (HYDRAVIPER - TR-A) and UCB-based validation budget allocation (HYDRAVIPER - VR-UCB).

Figure 7.10 shows the results of our ablation study on cooperative navigation (CN) and Cologne corridor (CC). On both environments and at both budget levels, centralised-Q resampling outperforms the IVIPER resampling scheme, although the gap in performance is less pronounced for the Cologne corridor due to environmental randomness. This result suggests that sampling the training dataset independently for each agent, instead of according to team performance, is insufficient for achieving coordinated behaviour in the resulting students. Meanwhile, removing the budget allocation methods degrades the performance of HYDRAVIPER. Having either one of the budget allocation methods is generally sufficient to improve HYDRAVIPER's reward, except in one case: for cooperative navigation at the low budget level, HYDRAVIPER performs worse when only one budget allocation mechanism is present. Meanwhile, for the Cologne corridor at the low budget level, the variance in HYDRAVIPER's reward is large both when *only* centralised-Q resampling is present, and also when it is *removed*. These results suggest that the primary benefit of the two rollout budget allocation mechanisms is to stabilise HYDRAVIPER's learning process, especially in the low budget setting when extracting the most information from each rollout is critical.

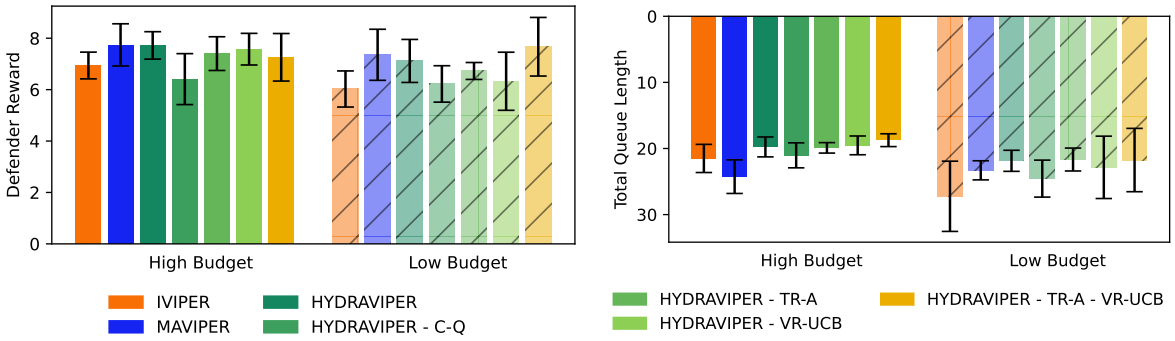


Figure 7.11: Ablation of HYDRAVIPER on the physical deception (PD, left, higher rewards are better) and Ingolstadt corridor (IC, right, lower rewards are better) environments.

7.6 Conclusion and Future Work

In this chapter, I introduced a new DT-based interpretable MARL method, HYDRAVIPER. HYDRAVIPER addresses several limitations of prior multi-agent methods in the VIPER framework: (1) it improves performance by using a joint dataset resampling scheme based on team Q-values, and (2) it improves computational efficiency by adaptively allocating fixed budgets of environment interactions for training and validation, as well as by dividing agents into jointly-trained teams. Based on experiments in benchmark environments for multi-agent coordination and traffic signal control, I showed that HYDRAVIPER achieves performance comparable with MAVIPER (a centralised method) and even neural network experts, all within a runtime less than IVIPER (a decentralised method). I also demonstrated HYDRAVIPER's sample efficiency in its ability to retain a similar level of performance using a fraction of the environment interactions.

Through my experiments in the Ingolstadt corridor environment, I scaled up the VIPER framework to seven agents. To my knowledge, this is the largest team of coordinated agents to which interpretable MARL has been applied thus far. However, environments based on real-world domains can have many more agents than the environments that I studied. For instance, the review of Noaeen et al. [274] showed that TSC environments of dozens or even hundreds of agents are used in the RL literature. At the extreme, Chen et al. [54] used parameter-shared MPLight policies as controller agents for an extremely large simulation of 2 510 traffic lights. My agent clustering approach shows promise in scaling up to larger environments while retaining performance comparable to that of expert policies. In scaling up, other methods of assigning edge weights e_{ij} to agent pairs could be developed to leverage structure present in MARL environments, including weights based on observation-action trajectories. I envision that the flexibility of the HYDRAVIPER framework will allow it to adapt to characteristics of different MARL environments while maintaining Pareto optimality in the performance-computational efficiency tradeoff.

Chapter 8

End of the Beginning

A Closing Vision of Deploying AI Systems for Transportation

Bibliography

[1] Montasir Abbas, Darcy Bullock, and Larry Head. 2001. Real-Time Offset Transitioning Algorithm for Coordinating Traffic Signals. *Transportation Research Record* 1748 (2001), 26–39.

[2] Monireh Abdoos and Ana L.C. Bazzan. 2021. Hierarchical traffic signal optimization using reinforcement learning and traffic prediction with long-short term memory. *Expert Systems with Applications* 171 (2021), 114580.

[3] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. 2019. Differentiable Convex Optimization Layers. In *Proceedings of the 33nd Conference on Neural Information Processing Systems (NeurIPS '19)*. NeurIPS, Vancouver, Canada, 9562–9574.

[4] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. 2022. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. arXiv:2206.14651

[5] Sanjeevan Ahilan and Peter Dayan. 2019. Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning. arXiv:1901.08492

[6] Lucas N. Alegre, Ana L.C. Bazzan, and Bruno C. da Silva. 2021. Quantifying the impact of non-stationarity in reinforcement learning-based traffic signal control. *PeerJ Computer Science* 7 (2021), e575.

[7] Yasir Ali, Zuduo Zheng, Md. Mazharul Haque, Mehmet Yildirimoglu, and Simon Washington. 2020. Understanding the discretionary lane-changing behaviour in the connected environment. *Accident Analysis & Prevention* 137 (2020), 105463.

[8] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18)*. AAAI, New Orleans, USA, 2669–2678.

[9] Tal Altshuler, Yaniv Altshuler, Rachel Katoshevski, and Yoram Shiftan. 2019. Modeling and Prediction of Ride-Sharing Utilization Dynamics. *Journal of Advanced Transportation* 2019 (2019), 6125798.

[10] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC '18)*. IEEE, Pis-

cataway, USA, 2575–2582.

[11] Anthropic. 2025. *The Claude 3 Model Family: Opus, Sonnet, Haiku*. Technical Report. Anthropic. 1–42 pages.

[12] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47 (2002), 235–256.

[13] James Ault, Josiah P. Hanna, and Guni Sharon. 2020. Learning an Interpretable Traffic Signal Control Policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*. IFAAMAS, Auckland, NZ, 88–96.

[14] James Ault and Guni Sharon. 2021. Reinforcement learning benchmarks for traffic signal control. In *Proceedings of the 35th Conference on Neural Information Processing Systems, Datasets and Benchmarks Track (NeurIPS '21)*. NeurIPS, Virtual, 1–11.

[15] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Fine-Tuning Language Models from Human Preferences. arXiv:2108.07732

[16] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI '17)*. AAAI, San Francisco, USA, 1726–1734.

[17] Esmaeil Balal, Ruey Long Cheu, Thompson Gyan-Sarkodie, and Jessica Miramontes. 2014. Analysis of Discretionary Lane Changing Parameters on Freeways. *International Journal of Transportation Science and Technology* 3, 3 (2014), 277–296.

[18] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Yokohama, Japan, 1–16.

[19] Angelo Banse and Jakob Erdmann. 2025. *Demand — Definition of Vehicles, Vehicle Types, and Routes*. German Aerospace Centre. `https://sumo.dlr.de/docs/Demand/Introduction_to_demand_modelling_in_SUMO.html`

[20] Angelo Banse and Jakob Erdmann. 2025. *Introduction to SUMO Networks*. German Aerospace Centre. `https://sumo.dlr.de/docs/Networks/SUMO_Road_Networks.html`

[21] Angelo Banse and Jakob Erdmann. 2025. *SSM Device*. German Aerospace Centre. `https://sumo.dlr.de/docs/Simulation/Output/SSM_Device.html`

[22] Yue Bao, Guangzhi Zang, Hai Yang, Ziyou Gao, and Jiancheng Long. 2023. Mathematical modeling of the platform assignment problem in a ride-sourcing market with a third-party integrator. *Transportation Research Part B: Methodological* 178 (2023), 102833.

[23] Jaume Barceló. 2010. Models, Traffic Models, Simulation, and Traffic Simulation. In *Fundamentals of Traffic Simulation*, Jaume Barceló (Ed.). Springer, New York, USA, 1–62.

[24] Osbert Bastani. 2021. Safe Reinforcement Learning with Nonlinear Dynamics via Model

Predictive Shielding. In *Proceedings of the 2021 American Control Conference (ACC '21)*. ACM, New Orleans, USA, 3488–3494.

[25] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS '18)*. NeurIPS, Montréal, Canada, 2494–2504.

[26] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software* 67, 1 (2015), 1–48.

[27] Matthew Battifarano and Sean Qian. 2019. Predicting real-time surge pricing of ride-sourcing companies. *Transportation Research Part C: Emerging Technologies* 107 (2019), 444–462.

[28] Nelson Baza-Solares, Ruben Velasquez-Martínez, Cristian Torres-Bohórquez, Yerly Martínez-Estupiñán, and Cristian Poliziani. 2022. Traffic Simulation with Open-Source and Commercial Traffic Microsimulators: A Case Study. *Communications — Scientific Letters of the University of Zilina* 24, 2 (2022), E49–E62.

[29] Luca Bedogni, Marco Gramaglia, Andrea Vesco, Marco Fiore, Jérôme Härri, and Francesco Ferrero. 2015. The Bologna Ringway Dataset: Improving Road Network Conversion in SUMO and Validating Urban Mobility via Navigation Services. *IEEE Transactions on Vehicular Technology* 64, 12 (2015), 5464–5476.

[30] Emma Beede, Elizabeth Baylor, Fred Hersch, Anna Iurchenko, Lauren Wilcox, Paisan Ruamviboonsuk, and Laura M. Vardoulakis. 2020. A Human-Centered Evaluation of a Deep Learning System Deployed in Clinics for the Detection of Diabetic Retinopathy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. ACM, Honolulu, USA, 1–12.

[31] Nikhil Behari, Edwin Zhang, Yunfan Zhao, Aparna Taneja, Dheeraj Nagaraj, and Milind Tambe. 2024. A Decision-Language Model (DLM) for Dynamic Restless Multi-Armed Bandit Tasks in Public Health. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS '24)*. NeurIPS, Vancouver, Canada, 1–38.

[32] Thor Berger, Carl Benedikt Frey, Guy Levin, and Santosh Rao Danda. 2020. Uber happy? Work and well-being in the 'Gig Economy'. *Economic Policy* 34, 99 (2020), 429–477.

[33] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d.O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv:1912.06680

[34] Umang Bhatt, Javier Antorán, Yunfeng Zhang, Q. Vera Liao, Prasanna Sattigeri, Riccardo Fogliato, Gabrielle Melançon, Ranganath Krishnan, Jason Stanley, Omesh Tickoo, Lama Nachman, Rumi Chunara, Madhulika Srikumar, Adrian Weller, and Alice Xiang. 2021. Uncertainty as a Form of Transparency: Measuring, Communicating, and Using Uncertainty. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (AIES*

'21). ACM, Virtual, 401–413.

[35] Laura Bieker, Daniel Krajzewicz, AntonioPio Morra, Carlo Michelacci, and Fabio Cartolano. 2014. Traffic Simulation for All: A Real World Traffic Scenario from the City of Bologna. In *Proceedings of the 2014 SUMO Conference (SUMO '14)*. Springer, Berlin, Germany, 47–60.

[36] Kostas Bimpikis, Ozan Candogan, and Daniela Saban. 2019. Spatial Pricing in Ride-Sharing Networks. *Operations Research* 67, 3 (2019), 744–769.

[37] Klavdiya Bochenina, Anton Taleiko, and Laura Ruotsalainen. 2023. Simulation-Based Origin-Destination Matrix Reduction: A Case Study of Helsinki City Area. In *Proceedings of the 2023 SUMO User Conference (SUMO '23)*. SUMO, Berlin, Germany, 1–7.

[38] Rohit Bokade, Xiaoning Jin, and Christopher Amato. 2023. Multi-Agent Reinforcement Learning Based on Representational Communication for Large-Scale Traffic Signal Control. *IEEE Access* 11 (2023), 47646–47658.

[39] P Bonhard and M A Sasse. 2006. 'Knowing me, knowing you' — using profiles and social networking to improve recommender systems. *BT Technology Journal* 24, 3 (2006), 84–98.

[40] W. Brilon and T. Wietholt. 2013. Experiences with Adaptive Signal Control in Germany. *Transportation Research Record* 2356 (2013), 9–16.

[41] Noam Brown and Tuomas Sandholm. 2017. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2017), 418–424.

[42] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. 2011. Non-Local Means Denoising. *Image Processing On Line* 1 (2011), 208–212.

[43] Eleanor R. Burgess, Ivana Jankovic, Melissa Austin, Nancy Cai, Adela Kapuścińska, Suzanne T. Currie, J. Marc Overhage, Erika S. Poole, and Jofsh Kaye. 2023. Healthcare AI Treatment Decision Support: Design Principles to Enhance Clinician Adoption and Trust. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–19.

[44] Carrie J. Cai, Samantha Winter, David Steiner, Lauren Wilcox, and Michael Terry. 2019. "Hello AI": Uncovering the Onboarding Needs of Medical Practitioners for Human–AI Collaborative Decision-Making. *Proceedings of the ACM on Human-Computer Interaction* 3 (2019), 1–24.

[45] Carrie J. Cai, Samantha Winter, David Steiner, Lauren Wilcox, and Michael Terry. 2021. Onboarding Materials as Cross-functional Boundary Objects for Developing AI Assistants. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (CHI EA '21)*. ACM, Yokohama, Japan, 1–7.

[46] Sibin Cai, Jie Fang, and Mengyun Xu. 2025. XLight: An interpretable multi-agent reinforcement learning approach for traffic signal control. *Expert Systems with Applications* 273 (2025), 126938.

[47] Wanling Cai, Yucheng Jin, and Li Chen. 2022. Impacts of Personal Characteristics on User Trust in Conversational Recommender Systems. In *Proceedings of the 2022 CHI Conference*

on Human Factors in Computing Systems (CHI '22). ACM, New Orleans, USA, 1–14.

[48] José António Capela Dias, Penousal Machado, and Francisco Câmara Pereira. 2013. Simulating the Impact of Drivers' Personality on City Transit. In *Proceedings of the 13th World Conference on Transport Research (WCTR '13)*. World Conference on Transport Research Society, Leeds, UK, 1–13.

[49] Federico Maria Cau, Hanna Hauptmann, Lucio Davide Spano, and Nava Tintarev. 2023. Effects of AI and Logic-Style Explanations on Users' Decisions under Different Levels of Uncertainty. *ACM Transactions on Interactive Intelligent Systems* 13, 4 (2023), 1–42.

[50] Ngai Keung Chan and Lee Humphreys. 2018. Mediatization of Social Space and the Case of Uber Drivers. *Media and Communication* 6, 2 (2018), 29–38.

[51] Cheng Chang, Siqi Wang, Jiawei Zhang, Jingwei Ge, and Li Li. 2024. LLMScenario: Large Language Model Driven Scenario Generation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54, 11 (2024), 6581–6594.

[52] Harshal A. Chaudhari, John W. Byers, and Evimaria Terzi. 2018. Putting Data in the Driver's Seat: Optimizing Earnings for On-Demand Ride-Hailing. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM '22)*. ACM, Marina Del Rey, USA, 90–98.

[53] Harshal A. Chaudhari, John W. Byers, and Evimaria Terzi. 2020. Learn to Earn: Enabling Coordination Within a Ride-Hailing Fleet. In *Proceedings of the 2020 IEEE International Conference on Big Data (Big Data '20)*. IEEE, Atlanta, USA, 1127–1136.

[54] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. 2020. Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*. AAAI, New York, USA, 3414–3421.

[55] Long Chen, Piyushimita (Vonu) Thakuriah, and Konstantinos Ampountolas. 2021. Short-Term Prediction of Demand for Ride-Hailing Services: A Deep Learning Approach. *Journal of Big Data Analytics in Transportation* 3 (2021), 175–195.

[56] M. Keith Chen, Peter E. Rossi, Judith A. Chevalier, and Emily Oehlsen. 2019. The Value of Flexible Work: Evidence from Uber Drivers. *Journal of Political Economy* 127, 6 (2019), 2735–2794.

[57] Rex Chen, Kathleen M. Carley, Fei Fang, and Norman Sadeh. 2023. Purpose in the Machine: Do Traffic Simulators Produce Distributionally Equivalent Outcomes for Reinforcement Learning Applications?. In *Proceedings of the 2023 Winter Simulation Conference (WSC '23)*. ACM, San Antonio, USA, 1842–1853.

[58] Rex Chen, Fei Fang, and Norman Sadeh. 2022. The Real Deal: A Review of Challenges and Opportunities in Moving Reinforcement Learning-Based Traffic Signal Control Systems Towards Reality. In *Proceedings of the 12th International Workshop on Agents in Traffic and Transportation (ATT '22)*. CEUR, Vienna, Austria, 1–21.

[59] Rex Chen, Ruiyi Wang, Norman Sadeh, and Fei Fang. 2025. Missing Pieces: How Do

Designs that Expose Uncertainty Longitudinally Impact Trust in AI Decision Aids? An In Situ Study of Gig Drivers. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency (FAccT '25)*. ACM, Athens, Greece, n. pag.

[60] Xiwen Chen, Hao Wang, Abolfazl Razi, Brendan Russo, Jason Pacheco, and John Roberts. 2023. Network-Level Safety Metrics for Overall Traffic Safety Assessment: A Case Study. *IEEE Access* 11 (2023), 17755–17778.

[61] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. 2019. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. In *Proceedings of the 33nd AAAI Conference on Artificial Intelligence (AAAI '19)*. AAAI, Honolulu, USA, 3387–3395.

[62] Hyesun Choung, Prabu David, and Arun Ross. 2023. Trust in AI and Its Role in the Acceptance of AI Technologies. *International Journal of Human-Computer Interaction* 39, 9 (2023), 1727–1739.

[63] Peter Christensen and Adam Osman. 2023. *The Demand for Mobility: Evidence from an Experiment with Uber Riders*. Working Paper 31330. NBER. 48 pages.

[64] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. 2019. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *Transportation Research Record* 21, 3 (2019), 1086–1095.

[65] Hyuck David Chung, Yue Maggie Zhou, and Christine Choi. 2022. When Uber Eats its Own Business, and That of its Competitors Too. *Academy of Management Proceedings* 2022, 1 (2022), 15263.

[66] Seung Youn Chyung, Megan Kennedy, and Ingrid Campbell. 2018. Evidence-Based Survey Design: The Use of Ascending or Descending Order of Likert-Type Response Options. *Performance Improvement* 57, 9 (2018), 9–16.

[67] Seung Youn Chyung, Katherine Roberts, Ieva Swanson, and Andrea Hankinson. 2017. Evidence-Based Survey Design: The Use of a Midpoint on the Likert Scale. *Performance Improvement* 56, 10 (2017), 15–23.

[68] Lara Codeca, Raphael Frank, and Thomas Engel. 2015. Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In *Proceedings of the 2015 Vehicular Networking Conference (VNC '15)*. IEEE, Kyoto, Japan, 1–8.

[69] Lara Codecá and Jérôme Härri. 2017. Towards multimodal mobility simulation of C-ITS: The Monaco SUMO traffic scenario. In *Proceedings of the 2017 Vehicular Networking Conference (VNC '17)*. IEEE, Turin, Italy, 97–100.

[70] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. 1998. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies* 6, 4 (1998), 271–288.

[71] Christopher Court-Dobson and Jon Lawson. 2024. *Can AI control traffic lights, signals and intersections?* Traffic Technology International. https://www.trafficTechnologytoday.com/features/

feature-should-ai-control-traffic-lights.html

[72] Marco Crespi, Andrea Ferigo, Leonardo Lucio Custode, and Giovanni Iacca. 2023. A population-based approach for multi-agent interpretable reinforcement learning. *Applied Soft Computing* 147 (2023), 110758.

[73] Colin Curtain. 2023. *QualCoder*. University of Tasmania. https://github.com/ccbogel/QualCoder/releases/tag/3.3

[74] Longchao Da, Chen Chu, Weinan Zhang, and Hua Wei. 2024. CityFlowER: An Efficient and Realistic Traffic Simulator with Embedded Machine Learning Models. In *Proceedings of the 2024 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD '24)*. ACM, Vilnius, Lithuania, 368–373.

[75] Longchao Da, Minquan Gao, Hao Mei, and Hua Wei. 2024. Prompt to transfer: sim-to-real transfer for traffic signal control with prompt learning. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI '24)*. AAAI, Vancouver, Canada, 82–90.

[76] Simon Danner, Matthias Pfromm, and Klaus Bengler. 2020. Does Information on Automated Driving Functions and the Way of Presenting It before Activation Influence Users' Behavior and Perception of the System? *Information* 11, 1 (2020), 54.

[77] Christopher M. Day and Darcy M. Bullock. 2011. Computational Efficiency of Alternative Algorithms for Arterial Offset Optimization. *Transportation Research Record* 2259 (2011), 37–47.

[78] Christopher M. Day and Darcy M. Bullock. 2011. *Optimization of Offsets and Cycle Length Using High Resolution Signal Event Data*. Working Paper SPR-3409. Joint Transportation Research Program. 36 pages.

[79] Taylor de O. Antes, Ana L.C. Bazzan, and Anderson Rocha Tavares. 2022. Information upwards, recommendation downwards: reinforcement learning with hierarchy for traffic signal control. *Procedia Computer Science* 201 (2022), 24–31.

[80] Arjan de Ruijter, Oded Cats, and Hans van Lint. 2024. Ridesourcing platforms thrive on socio-economic inequality. *Scientific Reports* 14 (2024), 7371.

[81] Augustin Degas, Mir Riyanul Islam, Christophe Hurter, Shaibal Barua, Hamidur Rahman, Minesh Poudel, Daniele Ruscio, Mobyen Uddin Ahmed, Shahina Begum, Md Aquif Rahman, Stefano Bonelli, Giulia Cartocci, Gianluca Di Flumeri, Gianluca Borghini, Fabio Babiloni, and Pietro Aricó. 2022. A Survey on Artificial Intelligence (AI) and eXplainable AI in Air Traffic Management: Current Trends and Development with Future Research Trajectory. *Applied Sciences* 12, 3 (2022), 1295.

[82] Nicola Dell, Vidya Vaidyanathan, Indrani Medhi, Edward Cutrell, and William Thies. 2012. "Yours is better!": participant response bias in HCI. In *Proceedings of the 2012 CHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, Austin, USA, 1321–1330.

[83] Delphine Delorme and Bongsob Song. 2001. *Human Driver Model for SmartAHS*. Technical Report UCB-ITS-PRR-2001-12. Institute of Transportation Studies, University of Cali-

fornia, Berkeley, Berkeley, USA. 1–49 pages.

[84] Richard W. Denney, Larry Head, and Kevin Spencer. 2008. *Signal timing under saturated conditions*. Technical Report FHWA-HOP-09-008. U.S. Department of Transportation Federal Highway Administration. 76 pages.

[85] Yining Di, Meng Xu, Zheng Zhu, Hai Yang, and Xiqun Chen. 2022. Analysis of ride-sourcing drivers' working Pattern(s) via spatiotemporal work slices: A case study in Hangzhou. *Transport Policy* 125 (2022), 336–351.

[86] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17, 83 (2016), 1–5.

[87] Django. 2023. *Django 4.1*. Django Software Foundation. `https://docs.djangoproject.com/en/4.1`

[88] Mark Dodgson. 1993. Learning, Trust, and Technological Collaboration. *Human Relations* 46, 1 (1993), 77–95.

[89] Alexander Domahidi, Eric Chu, and Stephen Boyd. 2013. ECOS: An SOCP solver for embedded systems. In *Proceedings of the 2013 European Control Conference (ECC '13)*. IEEE, Zurich, Switzerland, 3071–3076.

[90] Pedro M. d'Orey, Ricardo Fernandes, and Michel Ferreira. 2012. Empirical Evaluation of a Dynamic and Distributed Taxi-Sharing System. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC '12)*. IEEE, Anchorage, USA, 140–146.

[91] Finale Doshi-Velez and Been Kim. 2016. Towards A Rigorous Science of Interpretable Machine Learning. arXiv:1702.08608

[92] Richard Dowling, Alexander Skabardonis, and Vassili Alexiadis. 2004. *Traffic Analysis Toolbox, Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software*. U.S. Department of Transportation Federal Highway Administration.

[93] Jeff Druce, Michael Harradon, and James Tittle. 2021. Explainable artificial intelligence (XAI) for increasing user trust in deep reinforcement learning driven autonomous systems. arXiv:2106.03775

[94] Laura Eboli, Gabriella Mazzulla, and Giuseppe Pungillo. 2016. Combining speed and acceleration to define car users' safe or unsafe driving behaviour. *Transportation Research Part C: Emerging Technologies* 68 (2016), 113–125.

[95] Econolite. 2023. *Centracs Edaptive datasheet*. Document CNTRC-EDPTV 11.2023. Econolite Product Resource Library. 2 pages.

[96] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. 2013. Review of Microscopic Lane-Changing Models and Future Research Opportunities. *IEEE Transactions on Intelligent Transportation Systems* 14, 4 (2013), 1942–1956.

[97] Myungeun Eom and Byung-In Kim. 2020. The traffic signal control problem for intersections: a review. *European Transport Research Review* 12 (2020), 50.

[98] Jakob Erdmann. 2014. Lane-Changing Model in SUMO. In *Proceedings of the 2014 SUMO User Conference (SUMO '14)*. German Aerospace Center, Cologne, Germany, 77–88.

[99] Mohamed Essa and Tarek Sayed. 2020. Self-learning adaptive traffic signal control for real-time safety optimization. *Accident Analysis & Prevention* 146 (2020), 105713.

[100] Euthenics and TranSystems. 2023. *Preliminary Feasibility Study — Cuyahoga/Medina Traffic Study*. Technical Report PID 116069. City of Strongsville. 1–759 pages.

[101] Karim Fadhloun, Hesham Rakha, Amara Loulizi, and Abdessattar Abdelkef. 2015. Vehicle Dynamics Model for Estimating Typical Vehicle Accelerations. *Transportation Research Record* 2491 (2015), 61–71.

[102] Nahid Parvez Farazi, Bo Zou, Tanvir Ahamed, and Limon Barua. 2021. Deep reinforcement learning in transportation research: A review. *Transportation Research Interdisciplinary Perspectives* 11 (2021), 100425.

[103] K. J. Kevin Feng, Tony W. Li, and Amy X. Zhang. 2023. Understanding Collaborative Practices and Tools of Professional UX Practitioners in Software publishers. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–20.

[104] Andrea Ferrario and Michele Loi. 2022. How Explainability Contributes to Trust in AI. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. ACM, Seoul, South Korea, 1457–1466.

[105] Andres Ferraro, Dietmar Jannach, and Xavier Serra. 2020. Exploring Longitudinal Effects of Session-based Recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys '20)*. ACM, Virtual, 474–479.

[106] Figma, Inc. n.d.. *Figma*. Figma, Inc. `https://figma.com`

[107] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18)*. AAAI, New Orleans, USA, 2974–2982.

[108] Deepeka Garg, Maria Chli, and George Vogiatzis. 2018. Deep Reinforcement Learning for Autonomous Traffic Light Control. In *Proceedings of the 2018 3rd International Conference on Intelligent Transportation Engineering (ICITE '18)*. IEEE, Singapore, 214–218.

[109] Deepeka Garg, Maria Chli, and George Vogiatzis. 2019. A Deep Reinforcement Learning Agent for Traffic Intersection Control Optimization. In *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC '19)*. IEEE, Auckland, NZ, 4222–4229.

[110] Deepeka Garg, Maria Chli, and George Vogiatzis. 2019. Traffic3D: A Rich 3D-Traffic Environment to Train Intelligent Agents. In *Proceedings of the 19th International Conference on Computational Science (ICCS '19)*. Springer, New York, USA, 749–755.

[111] Deepeka Garg, Maria Chli, and George Vogiatzis. 2020. Multi-Agent Deep Reinforcement Learning for Traffic optimization through Multiple Road Intersections using Live Camera Feed. In *Proceedings of the 2020 IEEE Intelligent Transportation Systems Conference (ITSC*

'20). IEEE, Rhodes, Greece, 1–8.

[112] Deepeka Garg, Maria Chli, and George Vogiatzis. 2022. Fully-Autonomous, Vision-based Traffic Signal Control: from Simulation to Reality. In *Proceedings of the 21th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '22)*. IFAAMAS, Auckland, NZ, 454–462.

[113] Nikhil Garg and Hamid Nazerzadeh. 2022. Driver Surge Pricing. *Management Science* 68, 5 (2022), 3219–3235.

[114] Timothy Geary and David Danks. 2019. Balancing the Benefits of Autonomous Vehicles. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (AIES '19)*. ACM, Honolulu, USA, 1–6.

[115] Steven R. Gehrke. 2020. Uber service area expansion in three major American cities. *Journal of Transport Geography* 86 (2020), 102752.

[116] Wade Genders and Saiedeh Razavi. 2018. Evaluating reinforcement learning state representations for adaptive traffic signal control. In *Proceedings of the 9th International Conference on Ambient Systems, Networks and Technologies (ANT '18)*. Procedia Computer Science, Porto, Portugal, 26–33.

[117] Douglas Gettman, Lili Pu, Tarek Sayed, and Steve Shelby. 2008. *Surrogate Safety Assessment Model and Validation: Final Report*. Technical Report FHWA-HRT-08-051. U.S. Department of Transportation Federal Highway Administration. 322 pages.

[118] David Gibson, Milton K. (Pete) Mills, and Doug Rekenthaler Jr. 1998. Staying in The Loop: The Search for Improved Reliability of Traffic Sensing Systems Through Smart Test Instruments. *Public Roads* 62, 2 (1998), 47–51.

[119] Thomas Krendl Gilbert, Nathan Lambert, Sarah Dean, Tom Zick, Aaron Snoswell, and Soham Mehta. 2023. Reward Reports for Reinforcement Learning. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society (AIES '23)*. ACM, Montréal, Canada, 84–130.

[120] Yang Miang Goh and Peter E.D. Love. 2012. Methodological application of system dynamics for evaluating traffic safety policy. *Safety Science* 50, 7 (2012), 1594–1605.

[121] Yaobang Gong, Mohamed Abdel-Aty, Jinghui Yuan, and Qing Cai. 2020. Multi-Objective reinforcement learning approach for improving safety at intersections with adaptive traffic signal control. *Accident Analysis & Prevention* 144 (2020), 105655.

[122] Robert L. Gordon and Warren Tighe. 2005. *Traffic Control Systems Handbook*. U.S. Department of Transportation Federal Highway Administration.

[123] Lance R. Grenzeback, William R. Reilly, Paul O. Roberts, and Joseph R. Stowers. 1990. Urban Freeway Gridlock Study: Decreasing the Effects of Large Trucks on Peak-Period Urban Freeway Congestion. *Transportation Research Record* 1256 (1990), 16–26.

[124] Ulrike Gretzel and Daniel R. Fesenmaier. 2006. Persuasion in Recommender Systems. *International Journal of Electronic Commerce* 11, 2 (2006), 81–100.

[125] Kathleen Griesbach, Adam Reich, Luke Elliott-Negri, and Ruth Milkman. 2019. Algorith-

mic Control in Platform Food Delivery Work. *Socius* 5 (2019), 1–15.

[126] Hankang Gu, Shangbo Wang, Xiaoguang Ma, Dongyao Jia, Guoqiang Mao, and Eng Gee Lim. 2024. Large-Scale Traffic Signal Control Using Constrained Network Partition and Adaptive Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* 25, 7 (2024), 7619–7632.

[127] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, and Jun Wang. 2024. A Review of Safe Reinforcement Learning: Methods, Theories, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 12 (2024), 11216–11235.

[128] Yin Gu, Kai Zhang, Qi Liu, Weibo Gao, Longfei Li, and Jun Zhou. 2024. π-Light: Programmatic Interpretable Reinforcement Learning for Resource-Limited Traffic Signal Control. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI '24)*. AAAI, Vancouver, Canada, 21107–21115.

[129] Harish Guda and Upender Subramanian. 2019. Your Uber Is Arriving: Managing On-Demand Workers Through Surge Pricing, Forecast Communication, and Worker Incentives. *Management Science* 65, 5 (2019), 1995–2014.

[130] Jiaying Guo, Saeedeh Ghanadbashi, Shen Wang, and Fatemeh Golpayegani. 2023. Urban Traffic Signal Control at the Edge: An Ontology-Enhanced Deep Reinforcement Learning Approach. In *Proceedings of the 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC '23)*. IEEE, Bilbao, Spain, 6027–6033.

[131] Suiming Guo, Chao Chen, Jingyuan Wang, Yaxiao Liu, Ke Xu, and Zhiwen Yu. 2019. ROD-Revenue: Seeking Strategies Analysis and Revenue Prediction in Ride-on-Demand Service Using Multi-Source Urban Data. *IEEE Transactions on Mobile Computing* 19, 9 (2019), 2202–2220.

[132] Suiming Guo, Chao Chen, Jingyuan Wang, Yaxiao Liu, Ke Xu, Daqing Zhang, and Dah Ming Chiu. 2018. A Simple but Quantifiable Approach to Dynamic Price Prediction in Ride-on-demand Services Leveraging Multi-source Urban Data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 1–24.

[133] Gurobi Optimization, LLC. 2025. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC. `https://www.gurobi.com`

[134] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. 2021. Learning to Walk in the Real World with Minimal Human Effort. In *Proceedings of the 4th Annual Conference on Robot Learning (CoRL '21)*. CoRL, London, UK, 1110–1120.

[135] Jonathan V. Hall and Alan B. Krueger. 2018. An Analysis of the Labor Market for Uber's Driver-Partners in the United States. *ILR Review* 71, 3 (2018), 705–732.

[136] Xu Han, Qiannan Yang, Xianda Chen, Zhenghan Cai, Xiaowen Chu, and Meixin Zhu. 2024. AutoReward: Closed-Loop Reward Design with Large Language Models for Autonomous Driving. *IEEE Transactions on Intelligent Vehicles* Early Access (2024), 1–13.

[137] Xiao Han, Xiangyu Zhao, Liang Zhang, and Wanyu Wang. 2023. Mitigating Action Hysteresis in Traffic Signal Control with Traffic Predictive Reinforcement Learning. In *Pro-*

ceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23). ACM, Long Beach, USA, 673–684.

[138] Chengzheng Hang, Zhenfei Liu, Yujing Wang, Caiyi Hu, Yuelong Su, and Zhenning Dong. 2019. Sharing diseconomy: impact of the subsidy war of ride-sharing companies on urban congestion. *International Journal of Logistics Research and Applications* 22, 5 (2019), 491–500.

[139] Benjamin V. Hanrahan, Ning F. Ma, and Chien Wen Yuan. 2017. The Roots of Bias on Uber. In *Proceedings of the 15th European Conference on Computer-Supported Cooperative Work (ECSCW '17)*. EUSSET, Sheffield, UK, 1–17.

[140] Xin-Jing Hao. 2024. *PPO-Continuous-Pytorch*. Shanghai Jiao Tong University. `https://github.com/XinJingHao/PPO-Continuous-Pytorch`

[141] Brian Yueshuai He, Qinhua Jiang, Haoxuan Ma, and Jiaqi Ma. 2024. Multi-Agent Multimodal Transportation Simulation for Mega-cities: Application of Los Angeles. *Procedia Computer Science* 238 (2024), 736–741.

[142] Suining He and Kang G. Shin. 2019. Spatio-Temporal Capsule-based Reinforcement Learning for Mobility-on-Demand Network Coordination. In *Proceedings of the 2019 World Wide Web Conference (WWW '19)*. ACM, New York, USA, 2806–2813.

[143] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. 2022. Collective eXplainable AI: Explaining Cooperative Strategies and Agent Contribution in Multiagent Reinforcement Learning With Shapley Values. *IEEE Computational Intelligence Magazine* 17, 1 (2022), 59–71.

[144] Corey Hill, Lily Elefteriadou, and Alexandra Kondyli. 2015. Exploratory Analysis of Lane Changing on Freeways Based on Driver Behavior. *Journal of Transportation Engineering* 141, 4 (2015), 1–11.

[145] Gayang Ho, Clémence Morlet, Teik Soon Looi, Evan Gwee, Joel Teo, Peisi Keg, and Alok Jain. 2018. *Artificial Intelligence in Mass Public Transit*. Technical Report. UITP Asia-Pacific Centre for Transport Excellence. 13 pages.

[146] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58 (1963), 13–30.

[147] Hannah Horner, Jennifer Pazour, and John E. Mitchell. 2021. Optimizing driver menus under stochastic selection behavior for ridesharing and crowdsourced delivery. *Transportation Research Part E: Logistics and Transportation Review* 153 (2021), 102419.

[148] Moinul Hossain, Mohamed Abdel-Aty, Mohammed A. Quddus, Yasunori Muromachi, and Soumik Nafis Sadeek. 2019. Real-time crash prediction models: State-of-the-art, design pathways and ubiquitous requirements. *Accident Analysis & Prevention* 124 (2019), 66–84.

[149] Shengyi Huang and Santiago Ontañón. 2022. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. arXiv:2006.14171

[150] Nathan Hunt, Nathan Fulton, Sara Magliacane, Trong Nghia Hoang, Subhro Das, and Ar-

mando Solar-Lezama. 2021. Verifiably safe exploration for end-to-end reinforcement learning. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (HSCC '21)*. ACM, Virtual, 1–11.

[151] Hyeonjun Hwang, Clifford Winston, and Jia Yan. 2020. *Measuring the Benefits of Ridesharing Services to Urban Travelers: The Case of The San Francisco Bay Area*. Working Paper 70. Hutchins Center. 18 pages.

[152] Shariq Iqbal and Fei Sha. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML '19)*. JMLR, Long Beach, USA, 2961–2970.

[153] Lakshmi Shankar Iyer. 2021. AI enabled applications towards intelligent transportation. *Transportation Engineering* 5 (2021), 100083.

[154] Maia Jacobs, Jeffrey He, Melanie F. Pradier, Barbara Lam, Andrew C. Ahn, Thomas H. McCoy, Roy H. Perlis, Finale Doshi-Velez, and Krzysztof Z. Gajos. 2021. Designing AI for Trust and Collaboration in Time-Constrained Medical Decisions: A Sociotechnical Lens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Yokohama, Japan, 1–14.

[155] Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. 2021. Formalizing Trust in Artificial Intelligence: Prerequisites, Causes and Goals of Human Trust in AI. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*. ACM, Chicago, USA, 624–635.

[156] Vindula Jayawardana, Anna Landler, and Cathy Wu. 2021. Mixed Autonomous Supervision in Traffic Signal Control. In *Proceedings of the 2021 IEEE 24th International Conference on Intelligent Transportation Systems (ITSC '21)*. IEEE, Indianapolis, USA, 1767–1773.

[157] Bokai Ji, Guangxia Li, and Gang Xiao. 2023. Enhancing the Interpretability of Deep Multi-agent Reinforcement Learning via Neural Logic Reasoning. In *ICANN '23*. Springer, Crete, Greece, 199–210.

[158] Jiun-Yin Jian, Ann M. Bisantz, and Colin G. Drury. 2000. Foundations for an Empirically Determined Scale of Trust in Automated Systems. *International Journal of Cognitive Ergonomics* 4, 1 (2000), 53–71.

[159] Qize Jiang, Minhao Qin, Hanyuan Zhang, Xinyu Zhang, and Weiwei Sun. 2024. BlindLight: High Robustness Reinforcement Learning Method to Solve Partially Blinded Traffic Signal Control Problem. *IEEE Transactions on Intelligent Transportation Systems* 25, 11 (2024), 16625–16641.

[160] Shan Jiang, Le Chen, Alan Mislove, and Christo Wilson. 2018. On Ridesharing Competition and Accessibility: Evidence from Uber, Lyft, and Taxi. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. ACM, Lyon, France, 863–872.

[161] Xia Jiang, Jian Zhang, and Bo Wang. 2022. Energy-efficient driving for adaptive traffic signal control environment via explainable reinforcement learning. *Applied Sciences* 12, 11 (2022), 5380.

[162] Junchen Jin and Xiaoliang Ma. 2019. A Multi-Objective Agent-Based Control Approach With Application in Intelligent Traffic Signal System. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3900–3912.

[163] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, Guobin Wu, and Jieping Ye. 2019. CoRide: Joint Order Dispatching and Fleet Management for Multi-Scale Ride-Hailing Platforms. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. ACM, Beijing, China, 1983–1992.

[164] Glenn Jocher and Jing Qiu. 2024. *YOLO11*. Technical Report. Ultralytics. v11.0.0, `https://github.com/ultralytics/ultralytics`.

[165] Min-Wook Kang, Moynur Rahman, and Joyoung Lee. 2020. Determination and Utilization of Dilemma Zone Length and Location for Safety Assessment of Rural High-Speed Signalized Intersections. *Transportation Research Record* 2674, 4 (2020), 272–280.

[166] Amir Hossein Karbasi, Hao Yang, and Saiedeh Razavi. 2024. Exploring the impact of traffic signal control and connected and automated vehicles on intersections safety: A deep reinforcement learning approach. In *Proceedings of the 103rd Annual Meeting of the Transportation Research Board (TRB '24)*. TRB, Washington, DC, USA, 1–19.

[167] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1998), 359–392.

[168] Dmitry Kazhdan, Zohreh Shams, and Pietro Lio. 2020. MARLeME: A Multi-Agent Reinforcement Learning Model Extraction Library. In *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN '20)*. IEEE, Glasgow, UK, 1–8.

[169] Hassan Ali Khan, Muhammad Shahzad, Hassan Iqbal, and Guoliang Jin. 2022. RMS: Removing Barriers to Analyze the Availability and Surge Pricing of Ridesharing Services. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, New Orleans, USA, 1–18.

[170] Zulqarnain H. Khattak, Michael D. Fontaine, and Richard A. Boateng. 2018. Evaluating the impact of adaptive signal control technology on driver stress and behavior using real-world experimental data. *Transportation Research Part F: Traffic Psychology and Behaviour* 58 (2018), 133–144.

[171] Jungkeun Kim, Marilyn Giroux, and Jacob C. Lee. 2021. When do you trust AI? The effect of number presentation detail on consumer trust and acceptance of AI recommendations. *Psychology & Marketing* 38 (2021), 1140–1155.

[172] Sunnie S. Y. Kim, Q. Vera Liao, Mihaela Vorvoreanu, Stephanie Ballard, and Jennifer Wortman Vaughan. 2024. "I'm Not Sure, But...": Examining the Impact of Large Language Models' Uncertainty Expression on User Reliance and Trust. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)*. ACM, Rio de Janeiro, Brazil, 822–835.

[173] Sunnie S. Y. Kim, Elizabeth Anne Watkins, Olga Russakovsky, Ruth Fong, and Andrés Monroy-Hernández. 2023. Humans, AI, and Context: Understanding End-Users' Trust in a Real-World Computer Vision Application. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23)*. ACM, Chicago, USA, 77–88.

[174] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems* 23, 6 (2021), 4909–4926.

[175] Lawrence A. Klein, Milton K. Mills, and David R.P. Gibson. 2006. *Traffic Detector Handbook: Third Edition — Volume II*. U.S. Department of Transportation Federal Highway Administration.

[176] Bart P. Knijnenburg, Niels J.M. Reijmer, and Martijn C. Willemsen. 2011. Each to His Own: How Different Users Call for Different Interaction Methods in Recommender Systems. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys '11)*. ACM, Chicago, USA, 3–13.

[177] Amy J. Ko, Thomas D. LaToza, and Margaret M. Burnett. 2015. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering* 20 (2015), 110–141.

[178] Rafal Kocielnik, Saleema Amershi, and Paul N. Bennett. 2019. Will You Accept an Imperfect AI? Exploring Designs for Adjusting End-user Expectations of AI Systems. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, Glasgow, UK, 1–14.

[179] Spencer C. Kohn, Ewart J. de Visser, Eva Wiese, Yi-Ching Lee, and Tyler H. Shaw. 2021. Measurement of Trust in Automation: A Narrative Review and Reference Guide. *Frontiers in Psychology* 12 (2021), 604977.

[180] Sherrie Y. X. Komiak and Izak Benbasat. 2006. The Effects of Personalization and Familiarity on Trust and Adoption of Recommendation Agents. *MIS Quarterly* 30, 4 (2006), 941–960.

[181] Behrad Koohy, Sebastian Stein, Enrico Gerding, and Ghaithaa Manla. 2022. Reward Function Design in Multi-Agent Reinforcement Learning for Traffic Signal Control. In *Proceedings of the 12th International Workshop on Agents in Traffic and Transportation (ATT '22)*. International Joint Conference on Artificial Intelligence, Vienna, Austria, 1–13.

[182] Peter Koonce, Lee Rodegerdts, Kevin Lee, Shaun Quayle, Scott Beaird, Cade Braud, Jim Bonneson, Phil Tarnoff, and Tom Urbanik. 2008. *Traffic Signal Timing Manual*. U.S. Department of Transportation Federal Highway Administration.

[183] Moritz Körber, Christian Gold, David Lechner, and Klaus Bengler. 2016. The influence of age on the take-over of vehicle control in highly automated driving. *Transportation Research Part F: Traffic Psychology and Behaviour* 39 (2016), 19–32.

[184] Daniel Krajzewicz, Georg Hertkorn, C. Rössel, and Peter Wagner. 2002. SUMO (Simulation

of Urban MObility) - an open-source traffic simulation. In *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM '02)*. SCS Europe, Dubai, UAE, 183–187.

[185] Hanna Krasowski, Jakob Thumm, Marlon Müller, Lukas Schäfer, Xiao Wang, and Matthias Althoff. 2023. Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking. *Transactions on Machine Learning Research* November 2023 (2023), 1–38.

[186] Stefan Krauß. 1998. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. Ph. D. Dissertation. German Aerospace Center, Cologne, Germany.

[187] Ashwin Kumar, Yevgeniy Vorobeychik, and William Yeoh. 2023. Using Simple Incentives to Improve Two-Sided Fairness in Ridesharing Systems. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS '23)*. AAAI, Prague, Czech Republic, 227–235.

[188] Neetesh Kumar, Sarthak Mittal, Vaibhav Garg, and Neeraj Kumar. 2022. Deep Reinforcement Learning-Based Traffic Light Scheduling Framework for SDN-Enabled Smart Transportation System. *IEEE Transactions on Intelligent Transportation Systems* 23, 3 (2022), 2411–2421.

[189] Johannes Kunkel, Tim Donkers, Lisa Michael, Catalin-Mihai Barbu, and Jürgen Ziegler. 2019. Let Me Explain: Impact of Personal and Impersonal Explanations on Trust in Recommender Systems. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, Scotland, UK, 1–12.

[190] Alexandra Kuznetsova, Per B. Brockhoff, and Rune H. B. Christensen. 2017. lmerTest Package: Tests in Linear Mixed Effects Models. *Journal of Statistical Software* 82, 13 (2017), 1–26.

[191] Min-Ah Kwak, Theo Arentze, Erik de Romph, and Soora Rasouli. 2012. Activity-based dynamic traffic modeling: Influence of population sampling fraction size on simulation error. In *Proceedings of the 13th International Conference on Travel Behaviour Research (IATBR '12)*. IATBR, Toronto, Canada, 1–17.

[192] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. 2010. Temporal Diversity in Recommender Systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, Geneva, Switzerland, 210–217.

[193] Ho Lee and Benjamin Coifman. 2011. Quantifying Loop Detector Sensitivity and Correcting Detection Problems on Freeways. *Journal of Transportation Engineering* 138, 7 (2011), 871–881.

[194] Ho Lee and Benjamin Coifman. 2012. Identifying chronic splashover errors at freeway loop detectors. *Transportation Research Part C: Emerging Technologies* 24 (2012), 141–156.

[195] John Lee and Neville Moray. 1992. Trust, control strategies and allocation of function in human-machine systems. *Ergonomics* 10 (1992), 1243–1270.

[196] Min Kyung Lee, Danyel Kusbit, Evan Metsky, and Laura Dabbish. 2015. Working with

Machines: The Impact of Algorithmic and Data-Driven Management on Human Workers. In *Proceedings of the 2015 CHI Conference on Human Factors in Computing Systems (CHI '15)*. ACM, Seoul, South Korea, 1–13.

[197] Lauren Leffer. 2024. *Can Google Make Stoplights Smarter?* Scientific American. https://www.scientificamerican.com/article/googles-project-green-light-uses-ai-to-take-on-city-traffic/

[198] Yanzhe (Murray) Lei, Stefanus Jasin, Jingyi Wang, Houtao Deng, and Jagannath Putrevu. 2020. Dynamic Workforce Acquisition for Crowdsourced Last-Mile Delivery Platforms. arXiv:3532844

[199] Catur Yudo Leksono and Tina Andriyana. 2012. *Roundabout Microsimulation using SUMO: A Case Study in Idrottsparken Roundabout, Norrköping, Sweden*. Master's thesis. Linköping University, Linköping.

[200] Jonas F. Leon, Francesca Giancola, Andrea Boccolucci, and Mattia Neroni. 2023. A Demand Modelling Pipeline for an Agent-Based Traffic Simulation of the City of Barcelona. In *Proceedings of the 2023 Winter Simulation Conference (WSC '23)*. ACM, San Antonio, USA, 1777–1782.

[201] Shuyang Li, Talha Azfar, and Ruimin Ke. 2024. ChatSUMO: Large Language Model for Automating Traffic Scenario Generation in Simulation of Urban MObility. *IEEE Transactions on Intelligent Vehicles* Early Access (2024), 1–12.

[202] Toby Jia-Jun Li, Yuwen Lu, Jaylexia Clark, Meng Chen, Victor Cox, Meng Jiang, Yang Yang, Tamara Kay, Danielle Wood, and Jay Brockman. 2022. A Bottom-Up End-User Intelligent Assistant Approach to Empower Gig Workers against AI Inequality. In *Proceedings of the 2022 Symposium on Human-Computer Interaction for Work (CHIWORK '22)*. ACM, Durham, USA, 1–10.

[203] Yuxi Li. 2018. Deep Reinforcement Learning. arXiv:1810.06339

[204] Yongfu Li and Dihua Sun. 2012. Microscopic car-following model for the traffic flow: the state of the art. *Journal of Control Theory and Applications* 10 (2012), 133–143.

[205] Yichen Li and Chicheng Zhang. 2022. On Efficient Online Imitation Learning via Classification. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS '22)*. NeurIPS, New Orleans, USA, 32383–32397.

[206] Ziru Li, Chen Liang, Yili Hong, and Zhongju Zhang. 2022. How Do On-demand Ridesharing Services Affect Traffic Congestion? The Moderating Role of Urban Compactness. *Production and Operations Management* 31, 1 (2022), 239–258.

[207] Zhenning Li, Hao Yu, Guohui Zhang, Shangjia Dong, and Cheng-Zhong Xu. 2021. Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning. *Transportation Research Part C: Emerging Technologies* 125 (2021), 103059.

[208] Yu Liang and Martijn C. Willemsen. 2022. Exploring the longitudinal effects of nudging on users' music genre exploration behavior and listening preferences. In *Proceedings of the 16th ACM Conference on Recommender Systems (RecSys '22)*. ACM, Seattle, USA, 3–13.

[209] Lyuchao Liao, Jierui Liu, Xinke Wu, Fumin Zou, Jengshyang Pan, Qi Sun, Shengbo Eben Li, and Maolin Zhang. 2020. Time Difference Penalized Traffic Signal Timing by LSTM Q-Network to Balance Safety and Capacity at Intersections. *IEEE Access* 8 (2020), 80086–80096.

[210] Mengqi Liao and S. Shyam Sundar. 2022. When E-Commerce Personalization Systems Show and Tell: Investigating the Relative Persuasive Appeal of Content-Based versus Collaborative Filtering. *Journal of Advertising* 51 (2022), 256–267.

[211] Ulrich Lichtenthaler. 2018. Substitute or Synthesis: The Interplay between Human and Artificial Intelligence. *Research-Technology Management* 61, 5 (2018), 12–14.

[212] In-Kyu Lim and Young-Jun Kweon. 2013. Identifying High-Crash-Risk Intersections: Comparison of Traditional Methods with the Empirical Bayes–Safety Performance Function Method. *Transportation Research Record* 2364, 1 (2013), 44–50.

[213] Todd Litman. 2023. *Autonomous Vehicle Implementation Predictions: Implications for Transport Planning*. Technical Report. Victoria Transport Policy Institute. 49 pages.

[214] Shan Liu and Hai Jiang. 2022. Personalized route recommendation for ride-hailing with deep inverse reinforcement learning and real-time traffic conditions. *Transportation Research Part E: Logistics and Transportation Review* 164 (2022), 102780.

[215] Ying Liu, Lei Liu, and Wei-Peng Chen. 2017. Intelligent Traffic Light Control Using Distributed Multi-agent Q Learning. In *Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC '17)*. IEEE, Yokohama, Japan, 1–8.

[216] Zichuan Liu, Yuanyang Zhu, and Chunlin Chen. 2023. NA2Q: Neural Attention Additive Model for Interpretable Multi-Agent Q-Learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML '23)*. JMLR, Honolulu, USA, 22539–22558.

[217] Zichuan Liu, Yuanyang Zhu, Zhi Wang, Yang Gao, and Chunlin Chen. 2024. MIXRTs: Toward Interpretable Multi-Agent Reinforcement Learning via Mixing Recurrent Soft Decision Trees. arXiv:2209.07225

[218] Silas Lobo, Stefan Neumeier, Evelio M. G. Fernandez, and Christian Facchi. 2020. InTAS - The Ingolstadt Traffic Scenario for SUMO. In *Proceedings of the 2020 SUMO User Conference (SUMO '20)*. SUMO, Berlin, Germany, 73–92.

[219] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS '17)*. NeurIPS, Long Beach, USA, 6379–6390.

[220] Alice Lu, Peter Frazier, and Oren Kislev. 2018. Surge Pricing Moves Uber's Driver Partners. arXiv:3180246

[221] Zhuoran Lu and Ming Yin. 2021. Human Reliance on Machine Learning Models When Performance Feedback is Limited: Heuristics and Risks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Yokohama, Japan, 1–16.

[222] Felipe Luyanda, Douglas Gettman, Larry Head, Steven Shelby, Darcy Bullock, and Pitu Mirchandani. 2003. ACS-Lite Algorithmic Architecture: Applying Adaptive Control System Technology to Closed-Loop Traffic Signal Control Systems. *Transportation Research Record* 1856, 1 (2003), 175–184.

[223] Hongyao Ma, Fei Fang, and David C. Parkes. 2022. Spatio-Temporal Pricing for Ridesharing Platforms. *Operations Research* 70, 2 (2022), 1025–1041.

[224] Haitong Ma, Changliu Liu, Shengbo Eben Li, Sifa Zheng, Wenchao Sun, and Jianyu Chen. 2024. Learn Zero-Constraint-Violation Safe Policy in Model-Free Constrained Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems* 36, 2 (2024), 2327–2341.

[225] Jiaqi Ma, Michael D. Fontaine, Fang Zhou, and Jia Hu. 2016. Estimation of Crash Modification Factors for an Adaptive Traffic-Signal Control System. *Journal of Transportation Engineering* 142, 12 (2016), 04016061.

[226] Jinming Ma and Feng Wu. 2020. Feudal Multi-Agent Deep Reinforcement Learning for Traffic Signal Control. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*. IFAAMAS, Auckland, NZ, 816–824.

[227] Ning F. Ma and Benjamin V. Hanrahan. 2019. Part-Time Ride-Sharing: Recognizing the Context in which Drivers Ride-Share and its Impact on Platform Use. *Proceedings of the ACM on Human-Computer Interaction* 3 (2019), 1–17.

[228] Ning F. Ma, Zheng Yao, Veronica A. Rivera, and Dongwook Yoon. 2022. "Brush it Off": How Women Workers Manage and Cope with Bias and Harassment in Gender-agnostic Gig Platforms. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, New Orleans, USA, 1–13.

[229] Ning F. Ma, Chien Wen Yuan, Moojan Ghafurian, and Benjamin V. Hanrahan. 2018. Using Stakeholder Theory to Examine Drivers' Stake in Uber. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, Montréal, Canada, 1–12.

[230] Zian Ma, Chengcheng Xu, Yuheng Kan, Maonan Wang, and Wei Wu. 2021. Adaptive Coordinated Traffic Control for Arterial Intersections based on Reinforcement Learning. In *Proceedings of the 2021 IEEE 24th International Conference on Intelligent Transportation Systems (ITSC '21)*. IEEE, Indianapolis, USA, 2562–2567.

[231] Carl A. MacCarley, Stephen L.M. Hockaday, Daniel Need, and Samuel Taff. 1992. Evaluation of Video Image Processing Systems for Traffic Detection. *Transportation Research Record* 1360 (1992), 46–49.

[232] Michal Maciejewski. 2010. A Comparison of Microscopic Traffic Flow Simulation Systems for an Urban Area. *Transport Problems* 5, 4 (2010), 27–38.

[233] Maria Madsen and Shirley Gregor. 2000. Measuring Human-Computer Trust. In *Proceedings of the 11th Australasian Conference on Information Systems (ACIS '00)*. AAIS, Brisbane, Australia, 6–8.

[234] Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popović. 2017. Where to Add Actions in Human-in-the-Loop Reinforcement Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI '17)*. AAAI, San Francisco, USA, 2322–2328.

[235] Fred L. Mannering and Scott S. Washburn. 2018. Traffic Control and Analysis at Signalized Intersections. In *Principles of Highway Engineering and Traffic Analysis*. Wiley, Hoboken, USA, 243–310.

[236] Gustav Markkula, Ola Benderius, Krister Wolff, and Mattias Wahde. 2012. A Review of Near-Collision Driver Behavior Models. *Human Factors* 54, 6 (2012), 1117–1143.

[237] Jijo Mathew, Jairaj Desai, Rahul Suryakant Sakhare, Woosung Kim, Howell Li, and Darcy M. Bullock. 2021. Big Data Applications for Managing Roadways. *Institute of Transportation Engineers Journal* 91, 2 (2021), 28–35.

[238] Roger C. Mayer, James H. Davis, and F. David Schoorman. 1995. An Integrative Model of publisheral Trust. *The Academy of Management Review* 20, 3 (1995), 709–734.

[239] Jay Mayfield. 2024. *FTC Takes Action to Stop Lyft from Deceiving Drivers with Misleading Earnings Claims*. Federal Trade Commission. https://www.ftc.gov/news-events/news/press-releases/2024/10/ftc-takes-action-stop-lyft-deceiving-drivers-misleading-earnings-claim

[240] Juan Medina, Madhav Chitturi, and Rahim Benekohal. 2010. Effects of fog, snow, and rain on video detection systems at intersections. *Transportation Letters* 2, 1 (2010), 1–12.

[241] Juan C. Medina and Rahim F. Benekohal. 2012. Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC '12)*. IEEE, Anchorage, USA, 596–601.

[242] Juan C. Medina, Hani Ramezani, and Rahim (Ray) F. Benekohal. 2013. Evaluation of Microwave Radar Vehicle Detectors at a Signalized Intersection Under Adverse Weather Conditions. *Transportation Research Record* 2356 (2013), 100–108.

[243] Hao Mei, Xiaoliang Lei, Longchao Da, Bin Shi, and Hua Wei. 2024. LibSignal: an open library for traffic signal control. *Machine Learning* 113 (2024), 5235–5271.

[244] Hao Mei, Junxian Li, Bin Shi, and Hua Wei. 2023. Reinforcement learning approaches for traffic signal control under missing data. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI '23)*. ACM, Macau, 2261–2269.

[245] Xutao Mei, Nijiro Fukushima, Bo Yang, Zheng Wang, Tetsuya Takata, Hiroyuki Nagasawa, and Kimihiko Nakano. 2023. Reinforcement Learning-Based Intelligent Traffic Signal Control Considering Sensing Information of Railway. *IEEE Sensors Journal* 23, 24 (2023), 31125–31136.

[246] Wei Miao, Yiting Deng, Wei Wang, Yongdong Liu, and Christopher S. Tang. 2023. The effects of surge pricing on driver behavior in the ride-sharing market: Evidence from a quasi-experiment. *Journal of Operations Management* 69, 5 (2023), 794–822.

[247] Paul G Michael, Frank C Leeming, and William O Dwyer. 2000. Headway on urban streets:

observational data and an intervention to decrease tailgating. *Transportation Research Part F: Traffic Psychology and Behaviour* 3 (2000), 55–64. Issue 2.

[248] Panos G. Michalopoulos. 1991. A real-time computer vision system for vehicle tracking and traffic surveillance. *IEEE Transactions on Vehicular Technology* 40, 1 (1991), 21–29.

[249] Dan Middleton and Ricky Parker. 2000. *Initial Evaluation of Selected Detectors to Replace Inductive Loops on Freeways*. Technical Report FHWA/TX-00/1439-7. Texas Department of Transportation. 1–88 pages.

[250] Dan Middleton, Ricky Parker, and Ryan Longmire. 2007. *Investigation of Vehicle Detector Performance and ATMS Interface*. Technical Report FHWA/TX-07/0-4750-2. Texas Department of Transportation. 1–144 pages.

[251] Vicente Milanés and Steven E. Shladover. 2014. Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data. *Transportation Research Part C: Emerging Technologies* 48 (2014), 285–300.

[252] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. 2024. Explainable Reinforcement Learning: A Survey and Comparative Review. *Comput. Surveys* 56, 7 (2024), 1–36.

[253] Stephanie Milani, Zhicheng Zhang, Nicholay Topin, Zheyuan Ryan Shi, Charles Kamhoua, Evangelos E. Papalexakis, and Fei Fang. 2022. MAVIPER: Learning Decision Tree Policies for Interpretable Multi-agent Reinforcement Learning. In *Proceedings of the 2022 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD '22)*. ACM, Grenoble, France, 251–266.

[254] Shahin Mirbakhsh and Mahdi Azizi. 2024. Adaptive Traffic Signal's Safety and Efficiency Improvement by MultiObjective Deep Reinforcement Learning Approach. *International Journal of Innovative Research in Multidisciplinary Education* 3, 7 (2024), 1245–1257.

[255] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

[256] Christoph Molnar. 2019. Taxonomy of Interpretability Methods. In *Interpretable Machine Learning*. Independent, Munich, Germany.

[257] Seungwuk Moon and Kyongsu Yi. 2008. Human driving data-based design of a vehicle adaptive cruise control algorithm. *Vehicle System Dynamics* 46, 8 (2008), 661–690.

[258] Sara Moridpour, Majid Sarvi, and Geoff Rose. 2010. Lane changing models: a critical review. *Transportation Letters* 2, 3 (2010), 157–173.

[259] Sara Moridpour, Majid Sarvi, Geoff Rose, and Euan Ramsay. 2008. Variables influencing lane changing behaviour of heavy vehicles. In *Proceedings of the 31st Australasian Transport Research Forum (ATRF '08)*. Australasian Transport Research Forum, Canberra, Australia, 1–15.

[260] Yoshinari Motokawa and Toshiharu Sugawara. 2023. Interpretability for Conditional Coordinated Behavior in Multi-Agent Reinforcement Learning. In *Proceedings of the 2023 International Joint Conference on Neural Networks (IJCNN '23)*. IEEE, Gold Coast, Australia, 1–8.

[261] Jian Mou and Jason F. Cohen. 2017. Trust and online consumer health service success: A longitudinal study. *Information Development* 33, 2 (2017), 169–189.

[262] Jian Mou, Dong-Hee Shin, and Jason Cohen. 2017. Understanding trust and perceived usefulness in the consumer acceptance of an e-service: a longitudinal investigation. *Behaviour & Information Technology* 36, 2 (2017), 125–139.

[263] Arthur Müller, Vishal Rangras, Georg Schnittker, Michael Waldmann, Maxim Friesen, Tobias Ferfers, Lukas Schreckenberg, Florian Hufen, Jürgen Jasperneite, and Marco Wiering. 2021. Towards Real-World Deployment of Reinforcement Learning for Traffic Signal Control. In *Proceedings of the 20th IEEE International Conference on Machine Learning and Applications (ICMLA '21)*. IEEE, Pasadena, USA, 507–514.

[264] Arthur Müller and Matthia Sabatelli. 2022. Safe and Psychologically Pleasant Traffic Signal Control with Reinforcement Learning using Action Masking. In *Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC '22)*. IEEE, Macau, China, 951—-958.

[265] Arthur Müller and Matthia Sabatelli. 2023. Bridging the Reality Gap of Reinforcement Learning based Traffic Signal Control using Domain Randomization and Meta Learning. In *Proceedings of the 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC '23)*. IEEE, Bilbao, Spain, 5271–5278.

[266] Ziaul Haque Munim, Mariia Dushenko, Veronica Jaramillo Jimenez, Mohammad Hassan Shakil, and Marius Imset. 2020. Big data and artificial intelligence in the maritime industry: a bibliometric review and future research directions. *Maritime Policy & Management* 47, 5 (2020), 577–597.

[267] Mohammad Naiseh, Reem S. Al-Mansoori, Dena Al-Thani, Nan Jiang, and Raian Ali. 2021. Nudging through Friction: An Approach for Calibrating Trust in Explainable AI. In *Proceedings of the 8th International Conference on Behavioral and Social Computing (BESC '21)*. IEEE, Virtual, 1–5.

[268] Mohammad Naiseh, Dena Al-Thani, Nan Jiang, and Raian Ali. 2023. How the different explanation classes impact trust calibration: The case of clinical decision support systems. *International Journal of Human-Computer Studies* 169 (2023), 102941.

[269] Satoru Nakagawa, Dean Kriellaars, Christine Blais, Jeannette Montufar, and Michelle M. Porter. 2006. *Speed and acceleration patterns of younger and older drivers*. Technical Report. University of Manitoba, Winnipeg, Canada. 1–14 pages.

[270] Mohammadreza Nazari, Afshin Oroojlooy, Martin Takáč, and Lawrence V. Snyder. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS '18)*. NeurIPS, Montréal, Canada, 9861–9871.

[271] Gim Huay Neo, Robin Riedel, and Evgeni Kochman. 2025. *Intelligent Transport, Greener Future: AI as a Catalyst to Decarbonize Global Logistics*. Technical Report. World Economic Forum. 33 pages.

[272] Daiheng Ni. 2020. Controllers and Detectors. In *Signalized Intersections: Fundamentals to Advanced Systems*. Springer, New York, USA, 179–209.

[273] Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. 2018. Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets. In *Proceedings of the 2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC '18)*. IEEE, Maui, USA, 877–883.

[274] Mohammad Noaeen, Atharva Naik, Liana Goodman, Jared Crebo, Taimoor Abrar, Zahra Shakeri Hossein Abad, Ana L.C. Bazzan, and Behrouz Far. 2022. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems with Applications* 199 (2022), 116830.

[275] Mojtaba Norouzi, Monireh Abdoos, and Ana L. C. Bazzan. 2021. Experience classification for transfer learning in traffic signal control. *The Journal of Supercomputing* 77 (2021), 780–795.

[276] Hao Yi Ong, Daniel Freund, and Davide Crapis. 2021. Driver Positioning and Incentive Budgeting with an Escrow Mechanism for Ride-Sharing Platforms. *INFORMS Journal on Applied Analytics* 51, 5 (2021), 373–390.

[277] Aoyu Pang, Maonan Wang, Yirong Chen, Man-On Pun, and Michael Lepech. 2024. Scalable Reinforcement Learning Framework for Traffic Signal Control Under Communication Delays. *IEEE Open Journal of Vehicular Technology* 5 (2024), 330–343.

[278] Saumya Pareek, Eduardo Velloso, and Jorge Goncalves. 2024. Trust Development and Repair in AI-Assisted Decision-Making during Complementary Expertise. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)*. ACM, Rio de Janeiro, Brazil, 546–561.

[279] Darsh Parekh, Nishi Poddar, Aakash Rajpurkar, Manisha Chahal, Neeraj Kumar, Gyanendra Prasad Joshi, and Woong Cho. 2022. A Review on Autonomous Vehicles: Progress, Methods and Challenges. *Electronics* 11, 14 (2022), 2162.

[280] Bhavik Pathak, Robert Garfinkel, Ram D. Gopal, Rajkumar Venkatesan, and Fang Yin. 2010. Empirical Analysis of the Impact of Recommender Systems on Sales. *Journal of Management Information Systems* 27, 2 (2010), 159–188.

[281] Konstantin F. Pilz, Lennart Heim, and Nicholas Brown. 2025. Increased Compute Efficiency and the Diffusion of AI Capabilities. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI '25)*. AAAI, Philadelphia, USA, 27582–27590.

[282] Snehal Prabhudesai, Leyao Yang, Sumit Asthana, Xun Huan, Q. Vera Liao, and Nikola Banovic. 2023. Understanding Uncertainty: How Lay Decision-makers Perceive and Interpret Uncertainty in Human-AI Decision Making. In *Proceedings of the ACM Conference on Intelligent User Interfaces 2023 (IUI '23)*. ACM, Sydney, Australia, 379–396.

[283] K.J. Prabuchandran, A.N. Hemanth Kumar, and Shalabh Bhatnagar. 2015. Decentralized learning for traffic signal control. In *Proceedings of the 7th International Conference on Communication Systems and Networks (COMSNETS '15)*. IEEE, Bangalore, India, 2529—-2534.

[284] Yiheng Qian, Tejaswi Polimetla, Thomas W. Sanchez, and Xiang Yan. 2024. How do transportation professionals perceive the impacts of AI applications in transportation? A latent class cluster analysis. arXiv:2401.08915

[285] Zhiwei (Tony) Qin, Xiaocheng Tang, Yan Jiao, Fan Zhang, Zhe Xu, Hongtu Zhu, and Jieping Ye. 2020. Ride-Hailing Order Dispatching at DiDi via Reinforcement Learning. *INFORMS Journal on Applied Analytics* 50, 5 (2020), 272–286.

[286] Anjie Qiu, Prapul Alemada Sathish, Donglin Wang, and Hans D. Schotten. 2024. Advanced Traffic Demand Generation in SUMO: ML-based Prediction of Flow Rate based on Real-world Measured Datasets. In *Proceedings of the 2024 IEEE 99th Vehicular Technology Conference (VTC '24)*. IEEE, Singapore, 1–7.

[287] Filip Radlinski, Craig Boutilier, Deepak Ramachandran, and Ivan Vendrov. 2022. Subjective Attributes in Conversational Recommendation Systems: Challenges and Opportunities. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI '22)*. AAAI, Vancouver, Canada, 12287–12293.

[288] Md. Mokhlesur Rahman, Pooya Najaf, Milton Gregory Fields, and Jean-Claude Thill. 2022. Traffic congestion and its urban scale factors: Empirical evidence from American urban areas. *International Journal of Sustainable Transportation* 16, 5 (2022), 406–421.

[289] Ram Rajagopal and Pravin Varaiya. 2007. *Health of California's Loop Detector System*. Technical Report UCB-ITS-PRR-2007-13. California PATH Program. 59 pages.

[290] Nived Rajaraman, Lin F. Yang, Jiantao Jiao, and Kannan Ramchandran. 2020. Toward the Fundamental Limits of Imitation Learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS '20)*. NeurIPS, Vancouver, Canada, 2914–2924.

[291] Naveen Raman, Sanket Shah, and John P. Dickerson. 2021. Data-Driven Methods for Balancing Fairness and Efficiency in Ride-Pooling. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI '21)*. IJCAI, Montréal, Canada, 363–369.

[292] Varun Nagaraj Rao, Samantha Dalal, Eesha Agarwal, Dana Calacci, and Andrés Monroy-Hernández. 2025. Rideshare Transparency: Translating Gig Worker Insights on AI Platform Design to Policy. In *Proceedings of the 28th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW '25)*. ACM, Bergen, Norway, 1–49.

[293] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. 2022. Vehicular traffic simulation in the city of Turin from raw data. *IEEE Transactions on Mobile Computing* 21, 12 (2022), 4656–4666.

[294] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic value function factorisation for

deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML '18)*. JMLR, Stockholm, Sweden, 7234–7284.

[295] Nedal T. Ratrout and Imran Reza. 2014. Comparison of Optimal Signal Plans by Synchro & TRANSYT-7F Using PARAMICS — A Case Study. *Procedia Computer Science* 32 (2014), 372–379.

[296] Robert Rausch and David Miller. 2007. Advanced Transportation Controller Standards Update. *Institute of Transportation Engineers Journal* 77, 5 (2007), 29–33.

[297] Bijul Raveendran, Tom V. Mathew, and Nagendra R. Velaga. 2025. Impact of countdown timer on drivers' anticipation at the onset of yellow at signalized intersections. *Transportation Letters* 17, 2 (2025), 356–368.

[298] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. *Benchmarking Safe Exploration in Deep Reinforcement Learning*. Technical Report. OpenAI. 1–25 pages.

[299] Pouria Razzaghi, Amin Tabrizian, Wei Guo, Shulu Chen, Abenezer Taye, Ellis Thompson, Alexis Bregeon, Ali Baheri, and Peng Wei. 2022. A Survey on Reinforcement Learning in Aviation Applications. arXiv:2211.02147

[300] Yuqing Ren, Xuefei (Nancy) Deng, and K.D. Joshi. 2023. Unpacking Human and AI Complementarity: Insights from Recent Works. *ACM SIGMIS Database* 54, 3 (2023), 6–10.

[301] Duncan Rheingans-Yoo, Scott Duke Kominers, Hongyao Ma, and David C. Parkes. 2019. Ridesharing with driver location preferences. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI '19)*. ACM, Macau, 557–564.

[302] Avery Rhodes, Darcy M. Bullock, James R. Sturdevant, and Zachary Thomas Clark. 2005. *Evaluation of Stop Bar Video Detection Accuracy at Signalized Intersections*. Technical Report FHWA/IN/JTRP-2005/28. Joint Transportation Research Program, Indiana Department of Transportation and Purdue University. 1–418 pages.

[303] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2006. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, San Francisco, USA, 1135–1144.

[304] Brishen Rogers. 2015. The Social Costs of Uber. *University of Chicago Law Review* 82 (2015), 85–102.

[305] Aji Ronaldo and M. Taufiq Ismail. 2012. *Comparison of the two Micro-Simulation Software AIMSUN & SUMO for Highway Traffic Modelling*. Master's thesis. Linköping University, Linköping.

[306] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS '11)*. PMLR, Fort Lauderdale, USA, 627–635.

[307] Aaron M. Roth, Jing Liang, Ram Sriram, Elham Tabassi, and Dinesh Manocha. 2023. MSVIPER: Improved Policy Distillation for Reinforcement-Learning-Based Robot Navi-

gation. *Journal of the Washington Academy of Sciences* 109, 2 (2023), 27–58.

[308] Jingqing Ruan, Ziyue Li, Hua Wei, Haoyuan Jiang, Jiaming Lu, Xuantang Xiong, Hangyu Mao, and Rui Zhao. 2024. CoSLight: Co-optimizing Collaborator Selection and Decision-making to Enhance Traffic Signal Control. In *Proceedings of the 30th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '24)*. ACM, Barcelona, Spain, 2500–2511.

[309] Ziad A. Sabra, Douglas Gettman, R. David Henry, and Venkata Nallamothu. 2010. *Balancing Safety and Capacity in an Adaptive Signal Control System — Phase 1*. Technical Report FHWA-HRT-10-038. U.S. Department of Transportation Federal Highway Administration. 106 pages.

[310] Mustapha Saidallah, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. 2016. A Comparative Study of Urban Road Traffic Simulators. In *Proceedings of the 2016 International Conference on Civil, Transportation and Environment (ICCTE '16)*. MATEC, Guangzhou, China, 1–6.

[311] Mohammad Saifuzzaman and Zuduo Zheng. 2014. Incorporating human-factors in car-following models: A review of recent developments and research needs. *Transportation Research Part C: Emerging Technologies* 48 (2014), 379–403.

[312] Johnny Saldaña. 2015. First-Cycle Coding Methods. In *The Coding Manual for Qualitative Researchers* (3rd ed.). SAGE, San Diego, USA, 67–206.

[313] Johnny Saldaña. 2015. Second-Cycle Coding Methods. In *The Coding Manual for Qualitative Researchers* (3rd ed.). SAGE, San Diego, USA, 233–268.

[314] Iqbal H. Sarker. 2021. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science* 2 (2021), 420.

[315] R M Savithramma and R. Sumathi. 2023. Intelligent traffic signal controller for heterogeneous traffic using reinforcement learning. *Green Energy and Intelligent Transportation* 2, 6 (2023), 100124.

[316] J.R. Sayer, M.L. Mefford, P.S. Fancher, R.E. Ervin, and S.E. Bogard. 1997. An Experimental Design for Studying How Driver Characteristics Influence Headway Control. In *Proceedings of the 1st International Conference on Intelligent Transportation Systems (ITSC '97)*. IEEE, Piscataway, USA, 870–875.

[317] Bruce Schaller. 2021. Can sharing a ride make for less traffic? Evidence from Uber and Lyft and implications for cities. *Transport Policy* 102 (2021), 1–10.

[318] Nicolas Scharowski, Michaela Benk, Swen J. Kühne, Léane Wettstein, and Florian Brühlmann. 2023. Certification Labels for Trustworthy AI: Insights From an Empirical Mixed-Method Study. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23)*. ACM, Chicago, USA, 248–262.

[319] Max Schemmer, Niklas Kühl, Carina Benz, Andrea Bartos, and Gerhard Satzger. 2023. Appropriate Reliance on AI Advice: Conceptualization and the Effect of Explanations. In *Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI '23)*.

ACM, Sydney, Australia, 410–422.

[320] Lukas M Schmidt, Georgios Kontes, Axel Plinge, and Christopher Mutschler. 2021. Can you trust your autonomous car? Interpretable and verifiably safe reinforcement learning. In *Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV '21)*. IEEE, Nagoya, Japan, 171–178.

[321] Jakob Schoeffer, Niklas Kuehl, and Yvette Machowski. 2022. "There Is Not Enough Information": On the Effects of Explanations on Perceptions of Informational Fairness and Trustworthiness in Automated Decision-Making. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. ACM, Seoul, Korea, 1616–1628.

[322] David Schrank, Luke Albert, Kartikeya Jha, and Bill Eisele. 2023. *2023 Urban Mobility Report*. Technical Report. Texas A&M Transportation Institute. 80 pages.

[323] Lincoln V Schreiber, Lucas N Alegre, Ana LC Bazzan, and Gabriel de O Ramos. 2022. On the explainability and expressiveness of function approximation methods in RL-based traffic signal control. In *Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN '22)*. IEEE, Padua, Italy, 1–8.

[324] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347

[325] Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference (SciPy '10)*. Python in Science Conference, Austin, USA, 92–96.

[326] Ujjwal Sehrawat, Namit Sawhney, Tejaswini Yeleswarapu, and Nimmi Rangaswamy. 2021. The Everyday HCI of Uber Drivers in India: A Developing Country Perspective. In *Proceedings of the 24th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW '21)*. ACM, Virtual, 1–22.

[327] Mahmoud Selim, Amr Alanwar, Shreyas Kousik, Grace Gao, Marco Pavone, and Karl H. Johansson. 2022. Safe Reinforcement Learning Using Black-Box Reachability Analysis. *IEEE Robotics and Automation Letters* 7, 4 (2022), 10665–10672.

[328] Marc Semrau, Jakob Erdmann, Bernhard Friedrich, and René Waldmann. 2016. Simulation framework for testing ADAS in Chinese traffic situations. In *Proceedings of the 2016 SUMO User Conference (SUMO '16)*. German Aerospace Center, Cologne, Germany, 103–114.

[329] Susan Shaheen and Nelson Chan. 2016. Mobility and the Sharing Economy: Potential to Facilitate the First- and Last-Mile Public Transit Connections. *Built Environment* 42, 4 (2016), 573–588.

[330] Mahboubeh Shamsi, Abdolreza Rasouli Kenari, and Roghayeh Aghamohammadi. 2022. Reinforcement learning for traffic light control with emphasis on emergency vehicles. *The Journal of Supercomputing* 78, 4 (2022), 4911–4937.

[331] Jing Shang, Shunmei Meng, Jun Hou, Xiaoran Zhao, Xiaokang Zhou, and Rong Jiang. 2025. Graph-Based Cooperation Multiagent Reinforcement Learning for Intelligent Traffic

Signal Control. *IEEE Internet of Things Journal* 12, 10 (2025), 14362–14374.

[332] Steven G. Shelby, Darcy M. Bullock, Doug Gettman, Raj S. Ghaman, Ziad A. Sabra, and Nils Soyke. 2008. An Overview and Performance Evaluation of ACS Lite — A Low Cost Adaptive Signal Control System. In *Proceedings of the 87th Annual Meeting of the Transportation Research Board (TRB '08)*. TRB, Washington, DC, USA, 1–17.

[333] Tianyu Shi, Francois Xavier Devailly, Denis Larocque, and Laurent Charlin. 2024. Improving the Generalizability and Robustness of Large-Scale Traffic Signal Control. *IEEE Open Journal of Intelligent Transportation Systems* 5 (2024), 2–15.

[334] Yang Shi, Zhenbo Wang, Tim J. LaClair, Chieh (Ross) Wang, Yunli Shao, and Jinghui Yuan. 2023. A Novel Deep Reinforcement Learning Approach to Traffic Signal Control with Connected Vehicles. *Applied Sciences* 13, 4 (2023), 2750.

[335] Zheyuan Ryan Shi, Claire Wang, and Fei Fang. 2020. Artificial Intelligence for Social Good: A Survey. arXiv:2001.01818

[336] Donghee Shin. 2020. How do users interact with algorithm recommender systems? The interaction of users, algorithms, and performance. *Computers in Human Behavior* 109 (2020), 106344.

[337] Donghee Shin. 2021. The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI. *International Journal of Human-Computer Studies* 146 (2021), 102551.

[338] Andrew Silva, Taylor Killian, Ivan Rodriguez Jimenez, Sung-Hyun Son, and Matthew Gombolay. 2020. Optimization Methods for Interpretable Differentiable Decision Trees in Reinforcement Learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS '20)*. PMLR, Palermo, Italy, 1855–1865.

[339] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.

[340] Anubha Singh, Patricia Garcia, and Silvia Lindtner. 2023. Old Logics, New Technologies: Producing a Managed Workforce on On-Demand Service Platforms. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–15.

[341] Ho Chit Siu, Jaime Peña, Edenna Chen, Yutai Zhou, Victor Lopez, Kyle Palko, Kimberlee Chang, and Ross Allen. 2021. Evaluation of human-AI teams for learned and rule-based agents in Hanabi. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS '21)*. NeurIPS, Virtual, 16183–16195.

[342] Venkatesh Sivaraman, Leigh A. Bukowski, Joel Levin, Jeremy M. Kahn, and Adam Perer. 2023. Ignore, Trust, or Negotiate: Understanding Clinician Acceptance of AI-Based Treat-

ment Recommendations in Health Care. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–18.

[343] Stephen Smith, Gregory Barlow, Xiao-Feng Xie, and Zack Rubinstein. 2013. Smart urban signal networks: Initial application of the SURTRAC adaptive traffic signal control system. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS '13)*. PKP, Rome, Italy, 434–442.

[344] Elizabeth Solberg, Magnhild Kaarstad, Maren H. Rø Eitrheim, Rossella Bisio, Kine Reegard, and Marten Bloch. 2022. A Conceptual Model of Trust, Perceived Risk, and Reliance on AI Decision Aids. *Group & publisher Management* 47, 2 (2022), 187–222.

[345] Raghavan Srinivasan, Craig Lyon, Bhagwant Persaud, Jongdae Baek, Frank Gross, Sarah Smith, and Carl Sundstrom. 2012. Crash Modification Factors for Changes to Left-Turn Phasing. *Transportation Research Record* 2279, 1 (2012), 108–117.

[346] Jeff C. Stanley and Stephen L. Dorton. 2023. Exploring Trust With the AI Incident Database. In *Proceedings of the 2023 Human Factors and Ergonomics Society Annual Meeting (HFES '23)*. ACM, Washington DC, USA, 489–494.

[347] Aleksandar Stevanonic. 2010. Chapter Six - Implementation Costs and Benefits. In *Adaptive Traffic Control Systems: Domestic and Foreign State of Practice*. NHCRP Synthesis, Vol. 403. TRB, Washington, DC, USA, 36–42.

[348] Aleksandar Stevanovic, Cameron Kergaye, and Peter T. Martin. 2009. SCOOT and SCATS: A Closer Look into Their Operations. In *Proceedings of the 88th Annual Meeting of the Transportation Research Board (TRB '09)*. TRB, Washington, DC, USA, 1–17.

[349] Aleksandar Stevanovic and Peter T. Martin. 2008. Split-Cycle Offset Optimization Technique and Coordinated Actuated Traffic Control Evaluated Through Microsimulation. *Transportation Research Record* 2080 (2008), 48–56.

[350] Erose Sthapit and Peter Björk. 2019. Sources of value co-destruction: Uber customer perspectives. *Tourism Review* 74, 4 (2019), 780–794.

[351] Rachel E. Stuck, Brittany E. Holthausen, and Bruce N. Walker. 2021. Chapter 8 - The role of risk in human-robot trust. In *Trust in Human-Robot Interaction*. Academic Press, Cambridge, UK, 179–194.

[352] Haoran Su, Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. 2022. EMV-Light: A Decentralized Reinforcement Learning Framework for Efficient Passage of Emergency Vehicles. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI '22)*. AAAI, Virtual, 4593–4601.

[353] Dazhi Sun, Leslie Dodoo, Andres Rubio, Harsha Kalyan Penumala, Michael Pratt, and Srinivasa Sunkari. 2012. *Synthesis study of Texas signal control systems: technical report*. Technical Report FHWA/TX-13/0-6670-1. Texas A&M Transportation Institute. 1–65 pages.

[354] Daniel (Jian) Sun and Alexandra Kondyli. 2010. Modeling Vehicle Interactions during Lane-Changing Behavior on Arterial Streets. *Computer-Aided Civil and Infrastructure En-*

gineering 25 (2010), 557–571.

[355] Yuan Sun, Magdalayna Drivas, Mengqi Liao, and S. Shyam Sundar. 2023. When Recommender Systems Snoop into Social Media, Users Trust them Less for Health Advice. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–14.

[356] Yuan Sun, Navid Salami Pargoo, Peter J. Jin, and Jorge Ortiz. 2024. Optimizing Autonomous Driving for Safety: A Human-Centric Approach with LLM-Enhanced RLHF. In *Companion of the 2024 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp Companion '24)*. ACM, New York, USA, 76–80.

[357] S. Shyam Sundar, Anne Oeldorf-Hirsch, and Qian Xu. 2008. The Bandwagon Effect of Collaborative Filtering Technology. In *Proceedings of the 2008 CHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, Florence, Italy, 3453–3458.

[358] Srinivasa Sunkari, Apoorba Bibeka, Nadeem Chaudhary, and Kevin Balke. 2019. *Impact of Traffic Signal Controller Settings on the Use of Advanced Detection Devices*. Technical Report FHWA/TX-18/0-6934-R1. Texas A&M Transportation Institute. 1–67 pages.

[359] Richard S. Sutton and Andrew G. Barto. 2018. Finite Markov Decision Processes. In *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, USA, 53–88.

[360] Richard S. Sutton and Andrew G. Barto. 2018. Policy Approximation. In *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, USA, 257–264.

[361] Richard S. Sutton and Andrew G. Barto. 2018. The Reinforcement Learning Problem. In *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, USA, 1–26.

[362] Richard S. Sutton and Andrew G. Barto. 2018. Temporal-Difference Learning. In *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, USA, 143–166.

[363] Pang Suwanaposee, Carl Gutwin, Zhe Chen, and Andy Cockburn. 2023. 'Specially For You' — Examining the Barnum Effect's Influence on the Perceived Quality of System Recommendations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–11.

[364] Meirav Taieb-Maimon and David Shinar. 2001. Minimum and Comfortable Driving Headways: Reality versus Perception. *Human Factors* 43, 1 (2001), 159–172.

[365] Kai Liang Tan, Anuj Sharma, and Soumik Sarkar. 2020. Robust Deep Reinforcement Learning for Traffic Signal Control. *Journal of Big Data Analytics in Transportation* 2, 3 (2020), 263–274.

[366] Zhi Ming Tan, Nikita Aggarwal, Josh Cowls, Jessica Morley, Mariarosaria Taddeo, and Luciano Floridi. 2021. The ethical debate about the gig economy: A review and critical analysis. *Technology and Society* 65 (2021), 101594.

[367] Leo Tasca. 2000. *A review of the literature on aggressive driving research*. Technical Report. Ontario Ministry of Transportation, Toronto, Canada. 1–25 pages.

[368] Larry W. Thomas. 2014. *Effect of MUTCD on Tort Liability of Government Transportation Agencies*. Legal Research Digest 63. National Cooperative Highway Research Program. 99

pages.

[369] David Edmigio Valdivieso Tituana, Sang Guun Yoo, and Roberto O. Andrade. 2022. Vehicle Counting using Computer Vision: A Survey. In *Proceedings of the 2022 IEEE 7th International Conference for Convergence in Technology (I2CT '22)*. IEEE, Pune, India, 1–7.

[370] Ernest Peter Todosiev. 1963. *The action point model of the vehicle-driver system*. Ph. D. Dissertation. The Ohio State University, Columbus, USA.

[371] Tomer Toledo, Haris N. Koutsopoulos, and Moshe Ben-Akiva. 2007. Integrated driving behavior modeling. *Transportation Research Part C: Emerging Technologies* 15, 2 (2007), 96–112.

[372] Richard Tomsett, Alun Preece, Dave Braines, Federico Cerutti, Supriyo Chakraborty, Mani Srivastava, Gavin Pearson, and Lance Kaplan. 2020. Rapid Trust Calibration through Interpretable and Uncertainty-Aware AI. *Patterns* 1, 4 (2020), 100049.

[373] Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. 2020. Iterative Bounding MDPs: Learning Interpretable Policies via Non-Interpretable Methods. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*. AAAI, New York, USA, 9923–9931.

[374] Manish Tripathy, Jiaru Bai, and H. Sebastian (Seb) Heese. 2022. Driver collusion in ride-hailing platforms. *Decision Sciences* 54, 4 (2022), 434–446.

[375] Takane Ueno, Yuto Sawa, Yeongdae Kim, Jacqueline Urakami, Hiroki Oura, and Katie Seaborn. 2022. Trust in Human-AI Interaction: Scoping Out Models, Measures, and Methods. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22)*. ACM, New Orleans, USA, 1–7.

[376] M. Rehmat Ullah, Khurram S. Khattak, Zawar H. Khan, Mushtaq A. Khan, Nasru Minallah, and Akhtar N. Khan. 2021. Vehicular Traffic Simulation Software: A Systematic Comparative Analysis. *Pakistan Journal of Engineering and Technology* 4, 1 (2021), 66–78.

[377] Sandesh Uppoor and Marco Fiore. 2011. Large-scale urban vehicular mobility for networking research. In *Proceedings of the 2011 IEEE Vehicular Networking Conference (VNC '11)*. IEEE, Amsterdam, Netherlands, 62–69.

[378] Sandesh Uppoor and Marco Fiore. 2012. A large-scale vehicular mobility dataset of the Cologne urban area. In *Proceedings of the 14th French Conference on Algorithms and Telecommunications (AlgoTel '12)*. HAL, Hérault, France, 1–4.

[379] Niels van Berkel and Vassilis Kostakos. 2021. Recommendations for Conducting Longitudinal Experience Sampling Studies. In *Advances in Longitudinal HCI Research*. Springer, Berlin, Germany, 59–78.

[380] Elise van der Pol and Frans A. Oliehoek. 2016. Coordinated Deep Reinforcement Learners for Traffic Light Control. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS '16)*. NeurIPS, Barcelona, Spain, 1–8.

[381] Marko Vasić, Andrija Petrović, Kaiyuan Wang, Mladen Nikolić, Rishabh Singh, and Sarfraz

Khurshid. 2022. MoËT: Mixture of Expert Trees and its application to verifiable reinforcement learning. *Neural Networks* 151 (2022), 34–47.

[382] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki amd Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575 (2019), 350–354.

[383] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. 2017. StarCraft II: A New Challenge for Reinforcement Learning. arXiv:1708.04782

[384] Peter Wagner. 2008. Action point models of human driving behaviour. In *Proceedings of the 2008 Traffic Simulation Workshop*. Institute of Highway Engineering and Transport Planning, Graz University of Technology, Graz, Austria, 1.

[385] Caroline Wang, Ishan Durugkar, Elad Liebman, and Peter Stone. 2023. DM^2: Decentralized Multi-Agent Reinforcement Learning via Distribution Matching. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI '23)*. AAAI, Washington, DC, USA, 11699–11707.

[386] Dakuo Wang, Liuping Wang, Zhan Zhang, Ding Wang, Haiyi Zhu, Yvonne Gao, Xiangmin Fan, and Feng Tian. 2021. "Brilliant AI Doctor" in Rural Clinics: Challenges in AI-Powered Clinical Decision Support System Deployment. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Yokohama, Japan, 1–18.

[387] Hai Wang, Yijie Yu, Yingfeng Cai, Xiaobo Chen Long Chen, and Qingchao Liu. 2019. A Comparative Study of State-of-the-Art Deep Learning Algorithms for Vehicle Detection. *IEEE Intelligent Transportation Systems Magazine* 11, 2 (2019), 82–95.

[388] Maonan Wang, Aoyu Pang, Yuheng Kan, Man-On Pun, Chung Shue Chen, and Bo Huang. 2024. LLM-Assisted Light: Leveraging Large Language Model Capabilities for Human-Mimetic Traffic Signal Control in Complex Urban Environments. arXiv:2403.08337

[389] Xiaoyu Wang, Baher Abdulhai, and Scott Sanner. 2023. A Critical Review of Traffic Signal Control and A Novel Unified View of Reinforcement Learning and Model Predictive Control Approaches for Adaptive Traffic Signal Control. In *Handbook on Artificial Intelligence and Transport*. Edward Elgar, Northampton, USA, Chapter 17, 482–532.

[390] Xinzhi Wang, Huao Li, Hui Zhang, Michael Lewis, and Katia Sycara. 2021. Explanation of Reinforcement Learning Model in Dynamic Multi-Agent System. arXiv:2008.01508

[391] Xinru Wang and Ming Yin. 2021. Are Explanations Helpful? A Comparative Study of the Effects of Explanations in AI-Assisted Decision-Making. In *Proceedings of the 26th International Conference on Intelligent User Interfaces (IUI '21)*. ACM, College Station, USA, 318–328.

[392] Xinru Wang and Ming Yin. 2023. Watch Out for Updates: Understanding the Efects of Model Explanation Updates in AI-Assisted Decision Making. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–19.

[393] Yinquan Wang, Jianjun Wu, Huijun Sun, Ying Lv, and Junyi Zhang. 2024. Promoting Collaborative Dispatching in the Ride-Sourcing Market With a Third-Party Integrator. *IEEE Transactions on Intelligent Transportation Systems* 25, 7 (2024), 6889–6901.

[394] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8 (1992), 279–292.

[395] Hua Wei, Chacha Chen, Kan Wu, Guanjie Zheng, Zhengyao Yu, Vikash Gayah, and Zhenhui Li. 2019. Deep Reinforcement Learning for Traffic Signal Control along Arterials. In *Proceedings of the 1st Workshop on Deep Reinforcement Learning for Knowledge Discovery (DRL4KDD '19)*. ACM, Anchorage, USA, 1–7.

[396] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. ACM, Anchorage, USA, 1290–1298.

[397] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. 2019. CoLight: Learning Network-level Cooperation for Traffic Signal Control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. ACM, Beijing, China, 1913–1922.

[398] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. 2019. A Survey on Traffic Signal Control Methods. arXiv:1904.08117

[399] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. 2021. Recent Advances in Reinforcement Learning for Traffic Signal Control: A Survey of Models and Evaluation. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 12–18.

[400] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, London, UK, 2496–2505.

[401] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS '22)*. NeurIPS, New Orleans, USA, 24824–24837.

[402] Keji Wei, Vikrant Vaze, and Alexandre Jacquillat. 2021. Transit Planning Optimization Under Ride-Hailing Competition and Traffic Congestion. *Transportation Science* 56, 3 (2021), 725–749.

[403] Tianhao Wei and Changliu Liu. 2022. Safe Control with Neural Network Dynamic Models. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference (L4DC '22)*. PMLR, Palo Alto, USA, 739–750.

[404] Herbert Weinblatt, Erik Minge, and Scott Petersen. 2013. Length-Based Vehicle Classification Schemes and Length Bin Boundaries. *Transportation Research Record* 2339, 1 (2013), 19–29.

[405] Xiao Wen, Yuanchang Xie, Liming Jiang, Yan Li, and Tingjian Ge. 2022. On the interpretability of machine learning methods in crash frequency modeling and crash modification factor development. *Accident Analysis & Prevention* 168 (2022), 106617.

[406] David Gray Widder, Laura Dabbish, James D. Herbsleb, Alexandra Holloway, and Scott Davidoff. 2021. Trust in Collaborative Automation in High Stakes Software Engineering Work: A Case Study at NASA. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Yokohama, Japan, 1–13.

[407] Rainer Wiedemann. 1974. Simulation des straßenverkehrsflußes. *Publications of the Institute for Transportation, University of Karlsruhe* 8 (1974), 1–42.

[408] M. Wiering, J. Vreeken, J. van Veenen, and A. Koopman. 2004. Simulation and optimization of traffic in a city. In *Proceedings of the 2004 Intelligent Vehicles Symposium (IV '04)*. IEEE, Parma, Italy, 453–458.

[409] Philipp Wintersberger, Frederica Janotta, Jakob Peintner, Andreas Löcken, and Andreas Riener. 2021. Evaluating feedback requirements for trust calibration in automated vehicles. *it — Information Technology* 63, 2 (2021), 1–12.

[410] Alex J Wood, Mark Graham, Vili Lehdonvirta, and Isis Hjorth. 2019. Good Gig, Bad Gig: Autonomy and Algorithmic Control in the Global Gig Economy. *Employment and Society* 33, 1 (2019), 56–75.

[411] Jingda Wu, Zhiyu Huang, Zhongxu Hu, and Chen Lv. 2023. Toward Human-in-the-Loop AI: Enhancing Deep Reinforcement Learning via Real-Time Human Guidance for Autonomous Driving. *Engineering* 21 (2023), 75–91.

[412] Lingtao Wu, Dominique Lord, and Yajie Zou. 2015. Validation of Crash Modification Factors Derived from Cross-Sectional Studies with Regression Models. *Transportation Research Record* 2514 (2015), 88–96.

[413] Qiang Wu, Jianqing Wu, Jun Shen, Bo Du, Akbar Telikani, Mahdi Fahmideh, and Chao Liang. 2022. Distributed agent-based deep reinforcement learning for large scale traffic signal control. *Knowledge-Based Systems* 241 (2022), 108304.

[414] Qingjun Wu, Hao Zhang, Zhen Li, and Kai Liu. 2019. Labor control in the gig economy: Evidence from Uber in China. *Journal of Industrial Relations* 61, 4 (2019), 574–596.

[415] Oskar Wysocki, Jessica Katharine Davies, Markel Vigo, Anne Caroline Armstrong, Dónal

Landers, Rebecca Lee, and André Freitas. 2023. Assessing the communication gap between AI models and healthcare professionals: Explainability, utility and trust in AI-driven clinical decision-making. *Artificial Intelligence* 316 (2023), 103839.

[416] Donghan Xie, Zhi Wang, Chunlin Chen, and Daoyi Dong. 2020. IEDQN: Information Exchange DQN with a Centralized Coordinator for Traffic Signal Control. In *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN '20)*. IEEE, Glasgow, UK, 1–8.

[417] Bingyu Xu, Yaowei Wang, Zhaozhi Wang, Huizhu Jia, and Zongqing Lu. 2021. Hierarchically and cooperatively learning traffic signal control. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*. AAAI, Virtual, 669–677.

[418] Fumeng Yang, Chloe Rose Mortenson, Erik Nisbet, Nicholas Diakopoulos, and Matthew Kay. 2024. In Dice We Trust: Uncertainty Displays for Maintaining Trust in Election Forecasts Over Time. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, Honolulu, USA, 1–24.

[419] Huan Yang, Han Zhao, Yu Wang, Guoqiang Liu, and Danwei Wang. 2022. Deep Reinforcement Learning Based Strategy For Optimizing Phase Splits in Traffic Signal Control. In *Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC '22)*. IEEE, Macau, 2329–2334.

[420] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. 2020. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*. IFAAMAS, Auckland, NZ, 1566–1574.

[421] Qisong Yang, Thiago D. Sim ao, Simon H. Tindemans, and Matthijs T. J. Spaan. 2023. Advanced Transportation Controller Standards Update. *Machine Learning* 112 (2023), 859–887.

[422] Qian Yang, Yuexing Hao, Kexin Quan, Stephen Yang, Yiran Zhao, Volodymyr Kuleshov, and Fei Wang. 2023. Harnessing Biomedical Literature to Calibrate Clinicians' Trust in AI Decision Support Systems. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–14.

[423] Weiran Yao and Sean Qian. 2020. Learning to Recommend Signal Plans under Incidents with Real-Time Traffic Prediction. *Transportation Research Record* 2674, 6 (2020), 45–59.

[424] Kok-Lim Alvin Yau, Junaid Qadir, Hooi Ling Khoo, Mee Hong Ling, and Peter Komisarczuk. 2017. A Survey on Reinforcement Learning Models and Algorithms for Traffic Signal Control. *Comput. Surveys* 50, 3 (2017), 34.

[425] Mobin Yazdani, Majid Sarvi, Saeed Asadi Bagloee, Neema Nassir, Jeff Price, and Hossein Parineh. 2023. Intelligent vehicle pedestrian light (IVPL): A deep reinforcement learning approach for traffic signal control. *Transportation Research Part C: Emerging Technologies* 149 (2023), 103991.

[426] Bingquan Yu, Jinqiu Guo, Qinpei Zhao, Jiangfeng Li, and Weixiong Rao. 2020. Smarter and

Safer Traffic Signal Controlling via Deep Reinforcement Learning. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, Virtual, 3345–3348.

[427] Haoxiang Yu, Vaskar Raychoudhury, and Shrawani Silwal. 2020. Dynamic Taxi Ride Sharing using Localized Communication. In *Proceedings of the 21st International Conference on Distributed Computing and Networking (ICDCN '20)*. ACM, Kolkata, India, 1–10.

[428] Kun Yu, Shlomo Berkovsky, Ronnie Taib, Jianlong Zhou, and Fang Chen. 2019. Do I trust my machine teammate? An investigation from perception to decision. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. ACM, Los Angeles, USA, 460–468.

[429] Jinghui Yuan, Mohamed Abdel-Aty, Ling Wang, Jaeyoung Lee, Xuesong Wang, and Rongjie Yu. 2018. Real-Time Crash Risk Analysis of Urban Arterials Incorporating Bluetooth, Weather, and Adaptive Signal Control Data. In *Proceedings of the 97th Annual Meeting of the Transportation Research Board (TRB '18)*. TRB, Washington, DC, USA, 1–8.

[430] Rui Yue, Guangchuan Yang, Yichen Zheng, Yuxin Tian, and Zong Tian. 2022. Effects of traffic signal coordination on the safety performance of urban arterials. *Computational Urban Science* 2, 3 (2022), 1–13.

[431] Mireia Yurrita, Tim Draws, Agathe Balayn, Dave Murray-Rust, Nava Tintarev, and Alessandro Bozzon. 2023. Disentangling Fairness Perceptions in Algorithmic Decision-Making: the Effects of Explanations, Human Oversight, and Contestability. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, Hamburg, Germany, 1–21.

[432] Renos Zabounidis, Joseph Campbell, Simon Stepputtis, Dana Hughes, and Katia P. Sycara. 2023. Concept Learning for Interpretable Multi-Agent Reinforcement Learning. In *Proceedings of the 7th Annual Conference on Robot Learning (CoRL '23)*. CoRL, Atlanta, USA, 1828–1837.

[433] Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. 2020. MetaLight: Value-Based Meta-Reinforcement Learning for Traffic Signal Control. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*. AAAI, New York, USA, 1153–1160.

[434] Jing Zeng, Jie Xin, Ya Cong, Jiancong Zhu, Yihao Zhang, Weihao Jiang, and Shiliang Pu. 2022. HALight: Hierarchical Deep Reinforcement Learning for Cooperative Arterial Traffic Signal Control with Cycle Strategy. In *Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC '22)*. IEEE, Macau, 479–485.

[435] Zheng Zeng. 2021. GraphLight: Graph-based Reinforcement Learning for Traffic Signal Control. In *Proceedings of the 6th International Conference on Computer and Communication Systems (ICCCS '21)*. IEEE, Las Vegas, USA, 645–650.

[436] Angie Zhang, Alexander Boltz, Jonathan Lynn, Chun-Wei Wang, and Min Kyung Lee. 2023. Stakeholder-Centered AI Design: Co-Designing Worker Tools with Gig Workers through Data Probes. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing*

Systems (CHI '23). ACM, Hamburg, Germany, 1–18.

[437] Angie Zhang, Alexander Boltz, Chun-Wei Wang, and Min Kyung Lee. 2022. Algorithmic Management Reimagined For Workers and By Workers: Centering Worker Well-Being in Gig Work. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, New Orleans, USA, 1–20.

[438] Dongping Zhang, Jason Hartline, and Jessica Hullman. 2024. Designing Shared Information Displays for Agents of Varying Strategic Sophistication. In *Proceedings of the 27st ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW '24)*. ACM, San José, Costa Rica, 1–34.

[439] Gongquan Zhang, Fangrong Chang, Helai Huang, and Zilong Zhou. 2024. Dual-Objective Reinforcement Learning-Based Adaptive Traffic Signal Control for Decarbonization and Efficiency Optimization. *Mathematics* 12, 13 (2024), 2056.

[440] Gongquan Zhang, Fangrong Chang, Jieling Jin, Fan Yang, and Helai Huang. 2024. Multi-objective deep reinforcement learning approach for adaptive traffic signal control system with concurrent optimization of safety, efficiency, and decarbonization at intersections. *Accident Analysis and Prevention* 199 (2024), 107451.

[441] Gongquan Zhang, Jieling Jin, Fangrong Chang, and Helai Huang. 2024. Real-time traffic conflict prediction at signalized intersections using vehicle trajectory data and deep learning. *International Journal of Transportation Science and Technology* In Press (2024), 1–15.

[442] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. In *Proceedings of the 2019 World Wide Web Conference (WWW '19)*. ACM, New York, USA, 3620–3624.

[443] Huichu Zhang, Chang Liu, Weinan Zhang, Guanjie Zheng, and Yong Yu. 2020. GeneraLight: Improving Environment Generalization of Traffic Signal Control via Meta Reinforcement Learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20)*. ACM, Virtual, 1783–1792.

[444] Jingjing Zhang, Gediminas Adomavicius, Alok Gupta, and Wolfgang Ketter. 2020. Consumption and Performance: Understanding Longitudinal Dynamics of Recommender Systems via an Agent-Based Simulation Framework. *Information Systems Research* 31, 1 (2020), 76–101.

[445] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*. Springer, New York, USA, 321–384.

[446] Lun Zhang, Shan Jiang, and Zheng Wang. 2017. Schedule-Driven Signal Priority Control for Modern Trams Using Reinforcement Learning. In *Proceedings of the 17th COTA International Conference of Transportation Professionals (CICTP '17)*. ASCE, Shanghai, China, 2122–2132.

[447] Qiaoning Zhang, Matthew L. Lee, and Scott Carter. 2022. You Complete Me: Human-

AI Teams and Complementary Expertise. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, New Orleans, USA, 1–28.

[448] Tingru Zhang, Da Tao, Xingda Qu, Xiaoyan Zhang, Jihong Zeng, Haoyu Zhu, and Han Zhu. 2020. Automated vehicle acceptance in China: Social influence and initial trust are key determinants. *Transportation Research Part C: Emerging Technologies* 112 (2020), 220–233.

[449] Yunchang Zhang and Jon Fricker. 2021. Investigating Smart Traffic Signal Controllers at Signalized Crosswalks: A Reinforcement Learning Approach. In *Proceedings of the 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS '21)*. IEEE, Heraklion, Greece, 1–6.

[450] Yifeng Zhang, Harsh Goel, Peizhuo Li, Mehul Damani, Sandeep Chinchali, and Guillaume Sartoretti. 2025. CoordLight: Learning Decentralized Coordination for Network-Wide Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* Early Access (2025), 1–16.

[451] Yunfeng Zhang, Q. Vera Liao, and Rachel K. E. Bellamy. 2020. Effect of confidence and explanation on accuracy and trust calibration in AI-assisted decision making. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAccT '20)*. ACM, Barcelona, Spain, 295–305.

[452] Yiran Zhang, Khoa Vo, Longchao Da, Tiejin Chen, Xiaoou Liu, and Hua Wei. 2025. Poster Abstract: Reproducible and Low-cost Sim-to-Real Environment for Traffic Signal Control. In *Proceedings of the 16th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS '25)*. ACM, Irvine, USA, 1–2.

[453] Pengqian Zhao, Yuyu Yuan, and Ting Guo. 2022. Extensible Hierarchical Multi-Agent Reinforcement-Learning Algorithm in Traffic Signal Control. *Applied Sciences* 12, 24 (2022), 12783.

[454] Weiye Zhao, Tairan He, and Changliu Liu. 2023. Probabilistic Safeguard for Reinforcement Learning Using Safety Index Guided Gaussian Process Models. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference (L4DC '23)*. PMLR, Philadelphia, USA, 783–796.

[455] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. 2021. Learning Phase Competition for Traffic Signal Control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. ACM, Beijing, China, 1963–1972.

[456] Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei, Zhengyao Yu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. Diagnosing Reinforcement Learning for Traffic Signal Control. arXiv:1905.04716

[457] Zuduo Zheng. 2014. Recent developments and research needs in modeling lane changing. *Transportation Research Part B: Methodological* 60 (2014), 16–32.

[458] Bin Zhou, Qishen Zhou, Simon Hu, Dongfang Ma, Sheng Jin, and Der-Horng Lee. 2024.

Cooperative Traffic Signal Control Using a Distributed Agent-Based Deep Reinforcement Learning With Incentive Communication. *IEEE Transactions on Intelligent Transportation Systems* 25, 8 (2024), 10147–10160.

[459] Meizi Zhou, Jingjng Zhang, and Gediminas Adomavicius. 2021. *Longitudinal Impact of Preference Biases on Recommender Systems' Performance*. Technical Report 2021-10. Kelley School of Business Research Paper.

[460] Xinyu Zhou, ZhuHua Liao, Yijiang Zhao, Yizhi Liu, and Aiping Yi. 2025. Ride-hailing pick-up area recommendation in a vehicle-cloud collaborative environment: a feature-aware personalized clustering federated learning approach. *Cluster Computing* 28 (2025), 32.

[461] Hong Zhu, Fengmei Sun, Keshuang Tang, Tianyang Han, and Junping Xiang. 2024. A Coordination Graph Based Framework for Network Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* 25, 10 (2024), 14298–14312.

[462] Yuanyang Zhu, Xiao Yin, and Chunlin Chen. 2022. Extracting decision tree from trained deep reinforcement learning in traffic signal control. *IEEE Transactions on Computational Social Systems* 10, 4 (2022), 1997–2007.

[463] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro von Werra. 2025. BigCodeBench: Benchmarking Code Generation with Diverse Function Calls and Complex Instructions. In *Proceedings of the 2025 International Conference on Learning Representations (ICLR '25)*. ICLR, Singapore, 1–55.

[464] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-Tuning Language Models from Human Preferences. arXiv:1909.08593