# *Mixed Autonomous Supervision in Traffic Signal Control*

# Mixed Autonomous Supervision in Traffic Signal Control

Vindula Jayawardana[1], Anna Landler[2] and Cathy Wu[3]

***Abstract*— Traffic signal control is a critical component for ensuring smooth traffic flows in city corridors. To this end, deep reinforcement learning (RL) agents have recently been proposed. These learned black-box policies may out-perform manually tuned policies on average, but they remain imperfect and come at a cost of how easy they are to interpret and supervise. These challenges hinder their adoption in real-world systems. To address the first challenge, this paper devises naturally interpretable decision tree policies that imitate expert deep neural network policies. To address supervision, we define a new formalization called Mixed Autonomous Supervision (MAS), which concerns integrating an imperfect policy into an existing supervision system. We propose a two-part supervision model with online automated supervision and offline human supervision to implement MAS. We present a novel blind spot detection algorithm for decision tree policies to encourage the safe transfer of control to an automated fail-safe policy (online supervision) and an interactive dashboard *DTLight* for offline human supervision. We show the decision tree policies are just as performant as the RL policies, and the proposed supervision model has a significant benefit in scenarios derived from real traffic situations.**

## I. INTRODUCTION

With increasing urbanization, traffic congestion is becoming an increasingly critical problem with many negative societal externalities, including environmental pollution and energy consumption, apart from its substantial impact on GDP [1]. Furthermore, studies show that a significant portion of commute travel time delays is caused at signalized intersections [2]. Therefore, optimizing traffic control at signalized intersections can have a significant positive impact on the prevailing congestion problem. In light of this, traffic signals are often closely monitored by traffic signal control engineers to avoid any abrupt disturbances to the traffic flow within corridors. Such monitoring falls under the broader task of traffic signal supervision, which includes designing, deploying, monitoring, and maintaining traffic signals.

Numerous studies have proposed optimized traffic signal control methods, which are widely used in modern cities [3, 4]. Recently, Deep Reinforcement Learning (DRL) has been used to control traffic signals due to its ability to directly learn complex strategies in dynamic environments [1]. However, such DRL models are inherently black-box and are not transparent in the way they make decisions. Nevertheless, transparent decision mechanisms can be required by regulations and are essential for the long-term supervision of these control policies. Hence, the successful real-world adoption of these black-box models comes at a cost of how much traffic engineers and other related stakeholders can understand, trust and supervise their functionality. For this reason, having interpretable and easy-to-supervise control policies is crucial for traffic signal control, a direction that has received limited attention.

Furthermore, it is practically impossible to produce simulation environments that accurately capture the complexities of the real world [5, 6]. Thus, control policies trained in simulations may lead to costly failures when deployed in the real world. For example, RL policies trained in simulation environments that assume all vehicles can be treated the same way may fail when faced with vehicles that require different treatments (e.g., emergency vehicles). Relatedly, practical limitations in training DRL models even within the simulation environment can cause policy *blind spots*.

To enable supervision of black-box DRL models in traffic signal control, this work proposes learning decision tree policies by imitating "expert" DRL models. Using decision tree policies is beneficial for two reasons: 1) decision trees are transparent in the way they make decisions, and 2) the easy-to-modify nature of decision trees facilitates long-term supervision. Nonetheless, such decision tree policies are imperfect and unsuitable for direct deployment in safety-critical settings such as traffic signal control.

This motivates us to consider the setting of Mixed Autonomous Supervision, in which both automated and human supervision may co-exist. We introduce a two-part supervision model suitable for MAS, which conducts both online and offline supervision to leverage complementarity in supervision, discussed in Section IV. Below we summarize the contributions of this paper, which to the best of our knowledge, were not addressed in previous work.

1) We leverage decision tree policies to enable the supervision of black-box RL policies in traffic signal control.
2) We formally define Mixed Autonomous Supervision.
3) For online supervision, we design a novel blind spot detection algorithm to encourage safety by transferring control to an external supervisory entity.
4) For offline supervision, we introduce *DTLight*, an interactive dashboard to gradually and safely increase the overall system's performance via manipulation of decision tree policies.
5) We validate the proposed MAS method in traffic signal control scenarios derived from real-world situations.

*This work was supported by the MIT-IBM Watson AI Lab

[1]MIT Laboratory for Information & Decision Systems and Department of Electrical Engineering and Computer Science, Cambridge, MA, 02139, USA vindula@mit.edu

[2]MIT Department of Civil and Environmental Engineering, Cambridge, MA, 02139, USA alandler@mit.edu

[3]MIT Laboratory for Information & Decision Systems, Department of Civil and Environmental Engineering and Institute of Data Systems and Society, Cambridge, MA, 02139, USA cathywu@mit.edu

## II. BACKGROUND

### A. Traffic Signal Control

The conventional methods for controlling signalized intersections can be categorized mainly into two categories. First, fixed-time signal plans designed using historical data are used. Such methods are often pre-configured and do not adapt to the prevailing traffic condition. Second, adaptive traffic signal control methods which can accommodate dynamic fluctuations of the traffic are used. Such methods are most suitable for intersections facing relatively high randomness in observed traffic. Webster, a classic method in the transportation field, proposes a cycle-based signal plan for individual intersections based on a given phase sequence [7]. Where sensors are available, Max-Pressure maximizes the relief of pressure between incoming and outgoing lanes [8]. SOTL develops a cycle-based, dynamic phase length traffic signal policy with additional demand responsive rules [9].

Recently, reinforcement learning has been used for traffic signal control. Wei et al. propose IntelliLight [10] which controls individual intersections using RL agents that do not consider neighbor information. PressLight [11], which uses the previous state-of-the-art Max-Pressure method to formulate the reward function, and CoLight [12], which employs graph representation learning, are a few other recent works. However, these policies are often hard to interpret. Recent attempts have also been made to develop interpretable traffic signal control policies [13]. However, the lack of focus on supervision, a practical use case of interpretability, is a main limitation of such works.

### B. Supervision

Supervision is a widely used technique in practice to avoid any costly mistakes and failures in production. In particular, systems that have been designed and developed in simulation environments can benefit more from close supervision to detect and respond to production-level failures. Ramakrishnan et al. [14] propose how learning about blind spots of agents and humans can be used to manage hand-off decisions when humans and agents jointly act in the real world. Similarly, Wray et al. [15] study Semi-Autonomous Systems (SAS) within the context of automated planning. It then proposes a technique for planning in the semi-autonomous driving domain to transfer control safely and smoothly between an agent and the human. In this work, we leverage the easy-to-interpret nature of decision tree policies with traffic signal control domain knowledge to propose a supervision model in which both automated and human supervision co-exits.

Designing, deploying, monitoring, and maintaining traffic signal control policies is a complex and gradual process that, in a broad sense, falls under the more general task of traffic signal supervision. Traffic control engineers traditionally perform such supervision at signal control centers. However, due to limited human resources available to monitor and manage often state-wide signalized intersections, traffic signal supervision has become a challenging, yet essential service for smooth traffic flows within a city. To assist traffic engineers with the supervision process, systems like Automated Traffic Signal Performance Measures (ATSPMs) [16] have emerged as suites of performance measures, data collection, and data analysis tools. In this work, we are concerned with the type of supervision required when decision tree policies devised using imperfect RL policies are deployed in real-world traffic corridors.

## III. PRELIMINARIES

### A. Reinforcement Learning

Reinforcement learning is a powerful computational approach for learning from interaction between an agent and an environment [17]. An RL environment is commonly formulated as a five-tuple Markov Decision Process (MDP) which can be denoted as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$. In such MDP, $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions, $p(s_{t+1}|s_t, a_t)$ is the transition probability of next state given current state and action, $r(s_t, a_t) \in \mathbb{R}$ is the reward for taking action $a_t$ at state $s_t$, and $\gamma \in [0, 1]$ a discounting factor. RL algorithms will then search for an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that will maximize the expected cumulative discounted reward over the MDP.

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, \pi\right] \quad (1)$$

A common approach to solve such a problem is to estimate the state-action value $Q(s, a)$.

$$Q(s, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a, \pi\right] \quad (2)$$

In MDPs with large state spaces, function approximation is used to approximate the state-action values as exact approaches often are not scalable [18].

### B. Viper Algorithm

In our work, we use the Viper algorithm [19] for training decision tree policies for controlling signalized intersections. Viper devises shallower decision trees based on the notion of critical states. A critical state $s$ is such that failing to take the optimal action at $s$ results in forfeiting all subsequent rewards. In contrast, non-critical states are the states in which multiple actions can lead to similar subsequent consequences, and the optimal action choice is not critical for the model's performance. Viper assesses the criticality of a state using importance measure $I(\cdot)$ as given in Equation 3. Further, it uses an expert model trained in the same environment as a teacher model to guide training a student decision tree model. Algorithm 1 outlines the complete algorithm. It takes as input a four-tuple $(S, A, P, R)$, a teacher model $\pi_E$, and the function $I(\cdot)$. Initially, it uses the teacher model to sample trajectories from the MDP, whereas the student model itself is used for sampling trajectories in the subsequent iterations. The distribution of states obtained after using policy $\pi$ for $\mathcal{T}$ steps on the given MDP is defined as $d^{\pi}$.

$$\mathrm{I(s)} = \max_{a \in \mathcal{A}} Q^{\pi}(s, a) - \min_{a \in \mathcal{A}} Q^{\pi}(s, a) \quad (3)$$

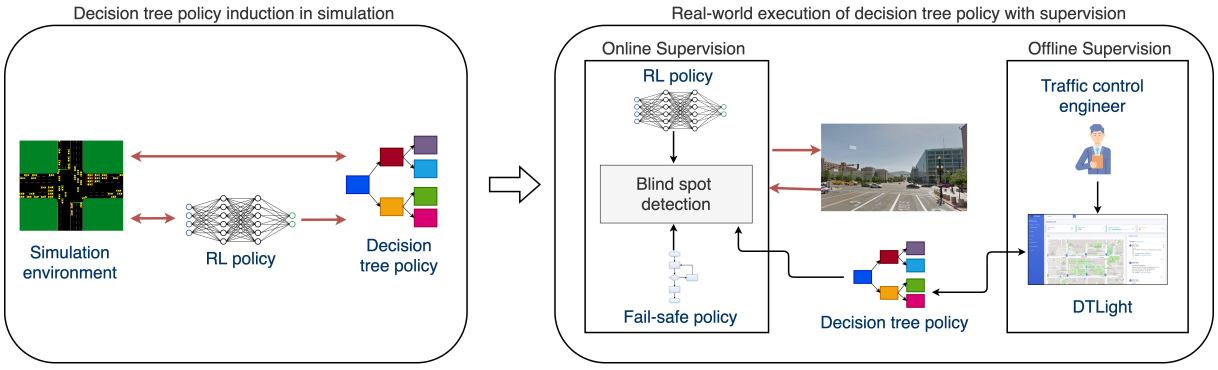



Fig. 1: Schematic overview of the proposed method

The importance measure $I(\cdot)$ is used to weigh the samples for training the student model. This sampling often leads to faster learning and shallower decision trees.

**Algorithm 1** Viper for producing decision tree policy $\pi_T$

**procedure** VIPER($(S, A, P, R)$, teacher $\pi_E$, importance function $I(\cdot)$, iterations $N$, number of trajectories $M$)
  Initialize dataset $D \leftarrow \emptyset$
  Initialize student policy $\pi_T^0 \leftarrow \pi_E$
  **for** $i : 1$ to $N$ **do**
    $M$ trajectories: $D_i \leftarrow \{(s, \pi_E(s)) \sim d^{\pi_T^{i-1}}\}$
    Aggregate: $D \leftarrow D \cup D_i$
    Sample: $\hat{D} \leftarrow \{(s, a) \sim p((s, a)) \propto I(s)\mathbb{I}[(s, a) \in D]\}$
    Train decision tree: $\pi_T^i \leftarrow train\_tree(\hat{D})$
  **end for**
  **return** Best policy $\pi_T \in \{\pi_T^1, \pi_T^2, \ldots, \pi_T^N\}$ selected via cross-validation

### C. Terminology

Below we define some terms used in this paper.

- *Control policy*: A stationary decision rule used to control a traffic signal.
- *Learned policy*: A data-driven control policy.
- *Defined policy*: A control policy that was designed by non-learning techniques.
- *RL Policy*: A learned policy trained using RL.
- *Decision tree policy*: A learned policy of the form of a decision tree trained to imitate an RL policy.
- *Fail-safe policy*: A defined policy designed to be safe and reasonably performant during execution.

## IV. METHOD

In this section, we elaborate on our method for devising and supervising decision tree policies for traffic signal control. The overall schematic overview of the proposed method is illustrated in Figure 1. We first describe devising decision tree policies, which will be instrumental for subsequent supervision tasks. Then, we formally define MAS and the two-part supervision model with online automated supervision and offline human supervision to implement MAS. We then describe a blind spot detection algorithm for implementing online supervision. Lastly, we introduce the interactive *DTLight* dashboard for offline supervision.

### A. Devising the Decision Tree Policy

To devise a decision tree policy, we employ an RL policy pre-trained in a traffic control environment as a teacher model. Then, using the Viper algorithm presented in Algorithm 1, we obtain a decision tree policy that minimizes its imitation error from the teacher model.

### B. Mixed Autonomous Supervision

**Definition IV.1** (Mixed Autonomous Supervision (MAS))**.** Mixed autonomy describes problems surrounding the gradual and complex integration of automation and AI into existing systems [20]. In the context of supervision, MAS involves integrating automation into existing supervision systems that were conventionally supervised only by humans. In such systems, automated and human supervision co-exist.

MAS aims to highlight the complementarity of supervision from automation and humans and thereby achieve the best of both worlds. In particular, synchronous human supervision at traffic control centers is limited, costly, and sub-optimal but leverages specialized training and common sense (including safety) that remain elusive for automated or manually designed agents. In traffic signal control, another form of human supervision exists, which takes the form of existing traffic signal controllers that have been manually designed and refined over decades. Such control policies are similarly expected to be safe and inexpensive to deploy but sub-optimal. On the other hand, automated supervision is inexpensive and "optimal" on specific, well-defined tasks but may be unsafe in out-of-distribution settings.

We propose that the three forms of supervision may be integrated to achieve the best of both worlds: 1) we leverage the inexpensive manually designed control policies for *online supervision*, to ensure the safe utilization of automated supervision, and 2) we leverage expensive human supervision periodically (*offline supervision*) to enable the performance improvement of the overall system.

### C. Online Supervision

Online supervision refers to synchronous supervision required to monitor when control should be transferred from

the decision tree policy to an external entity, e.g., to ensure safety. We formalize the online supervision problem by using the language of Semi-Autonomous Systems (SAS) [15], described in Section II-B. Whereas SAS considers human-agent collaboration, we adapt SAS for the online (autonomous) supervision context. Formally, we augment the MDP with a set of control policies.

**Definition IV.2** (Online Supervision)**.** Online Supervision is denoted by an extended Markov Decision Process $\langle \mathcal{M}, C, \mathcal{S}^+, \mathcal{A}^+, p^+, r^+ \rangle$ where,

- $\mathcal{M}$ is the underlying MDP of the environment.
- $C$ is a set of control policies.
- $\mathcal{S}^+$ is the set $\mathcal{S} \times C$ : an MDP state $s \in \mathcal{S}$ of $\mathcal{M}$ and the current control policy $c \in C$.
- $\mathcal{A}^+$ is the set $\mathcal{A} \times C$ : an MDP action $a \in \mathcal{A}$ of $\mathcal{M}$ and a desired next control policy $c \in C$.
- $p^+$ is the transition function $\mathcal{S}^+ \times \mathcal{A}^+ \to \Delta^{|\mathcal{S}^+|}$ which consists of control policy state transition function and transfer-of-control function as defined in Definition IV.3 and in Definition IV.4, respectively. Then, $p^+$ is given in Definition IV.5.
- $r^+$ is the reward function $\mathcal{S}^+ \times C \to \mathbb{R}$.

**Definition IV.3** (Control policy state transition function)**.** Control policy state transition function $p$ defines how a given control policy $c \in C$ will operate in the real world: $p : \mathcal{S} \times \mathcal{A} \to \Delta^{|\mathcal{S}|}$.

**Definition IV.4** (Transfer-of-control function)**.** Transfer-of-control function $\psi$ describes the result of attempting to transfer the control from one control policy to another control policy while operating in the real world: $\psi : \mathcal{S}^+ \times C \to \Delta^{|C|}$.

**Definition IV.5** (Online supervision state transition)**.** In the online supervision problem, given $s^+ = \langle s, c \rangle \in \mathcal{S}^+$, $a^+ = \langle a, c' \rangle \in \mathcal{A}^+$, and $\hat{s}^+ = \langle \hat{s}, \hat{c} \rangle \in \mathcal{S}^+$, online supervision state transition function is defined as,

$$p^+(s^+, a^+, \hat{s}^+) = \begin{cases} p(s, a, \hat{s}) & \text{if } c = c' = \hat{c} \\ p(s, a, \hat{s})\psi(s^+, c', \hat{c}) & c \neq c' \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In Equation 4, the first case corresponds to the instance that the current control policy is kept for the next state as well. In the second case, a new control policy is desired for operating in the next state. Finally, the third case denotes that control transfer is impossible without the desire to transfer.

**Definition IV.6** (Supervised policy)**.** The supervised policy is the control policy resulting from online supervision.

In this work, we specialize to deterministic transfer-of-control function and two control policies ($|C| = 2$): a decision tree policy and a fail-safe policy. We leverage existing traffic signal controllers as our fail-safe policies, assumed to be suboptimal but safe. In contrast, the decision tree policy is derived from an RL policy, assumed to be performant but potentially unsafe. We hypothesize that by designing an appropriate transfer-of-control function $\psi(\cdot)$, it is possible to achieve the best of both worlds.

### *D. Blind spot detection*

This section presents a novel blind spot detection algorithm, which serves as the transfer-of-control function $\psi(\cdot)$ between the fail-safe and decision tree policies. We justify the choice of the algorithm with intuitive arguments and defer theoretical analysis to future work.

**Assumptions**. For ease of discussion, we assume that the fail-safe policy is also in the form of a decision tree. Additionally, the action space is discrete, which is typically satisfied in traffic signal control. Both decision tree and fail-safe policies use the same input features.

We define a *blind spot* as a state for which the decision tree policy may fail to be safe. Intuitively, for any state, if the two policies *disagree* on the action to take, a suitable blind spot detection algorithm should trigger when any of the following conditions are not met:

1) *Decision structure*. The decision tree policy should agree with the general structure of the fail-safe policy, assumed to encode pertinent decisions regarding safety.
2) *High confidence*. The RL policy is confident about what action to take. Intuitively, the decision tree may override the fail-safe if it has visited the state often.
3) *Imitation consistency*. The decision tree policy should agree with the RL policy.

In such cases, it will be safe (though possibly less performant) to defer to the fail-safe policy.

**Algorithm 2** Decision tree blind spot detection

```
1:  procedure DETECT BLIND SPOTS(Decision tree policy T, RL policy E, fail-safe policy H, Observation o)
2:    Decision tree action a_T ← T(o)
3:    RL action a_E ← E(o)
4:    Fail-safe action a_H ← H(o)
5:    if a_T = a_H then
6:      blind spot ← False
7:    else
8:      if F*_o ≠ ∅ and F*_o ⊈ F^{T*}_o then
9:        blind spot ← True
10:     else
11:       Q ← obtain Q values for the o from E
12:       if max_a Q(o,a) − min_a Q(o,a) ≤ β then
13:         blind spot ← True
14:       else
15:         if a_T ≠ a_E then
16:           blind spot ← True
17:   return blind spot
```

Next, we formally elaborate on the blind spot detection algorithm for decision tree policies, presented in Algorithm 2. We start by obtaining proposed control actions for each policy (line 2-4). In lines 5-6, we check if the decision tree control action is different from the fail-safe policy. If so,

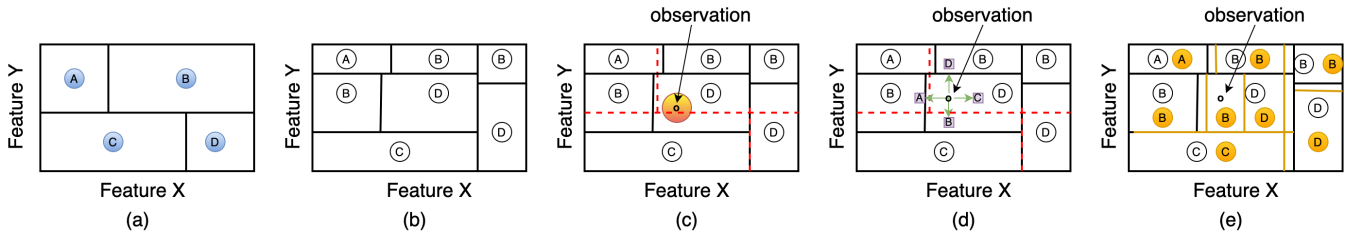



Fig. 2: Illustration of three sources of disagreements for identifying decision tree blind spots. **a)** Coarse-grained decision regions marked by decision boundaries of the fail-safe policy (each region has an action A, B, C or D); **b)** Fine-grained decision regions marked by decision boundaries of the learned decision tree policy; **c)** *Decision structure*: observation falls close to a decision boundary of the fail-safe policy as marked by the shaded circle but not in decision tree policy (red dashed lines denote the decision boundaries of the fail-safe policy); **d)** *High confidence*: decision tree policy is not confident in which action to be picked for the observation and all actions are equally preferable (arrow color intensity indicates the relative confidence); **e)** *Imitation consistency*: decision tree policy and RL policy disagree on the action (RL policy is presented in yellow). If any of the above conditions are met, we transfer the control to the fail-safe policy.

we must check for a possible blind spot by examining the above three conditions. In Figure 2, we illustrate the three conditions by a simplified example in which the state space consists of only two features, $X$ and $Y$.

*a) Decision structure:* To check for consistency in decision structure, we look at the features used by the fail-safe policy in its decision criterion for the given observation. We define a *critical feature* as a feature input where an $\pm\alpha$ change in input would yield a different decision criterion for the same observation. Let $F_o^*$ and $F_o^{T^*}$ be the set of critical features of observation $o$ for the fail-safe and decision tree policies, respectively. The critical features $F_o^*$ mark the decision boundaries of the fail-safe policy around the given observation and thus can be considered important features for guaranteeing safety. For the decision tree action to be considered, $F_o^* \subseteq F_o^{T^*}$ should be satisfied. That is, if the fail-safe policy uses a critical feature in its decision criterion, the decision tree policy should also consider it to be a critical feature (see Figure 2c in our example), or there may be an unsafe outcome. In practice, an engineer may choose to define the critical features $F_o^*$ directly, instead of using the above heuristic.

*b) High confidence:* Next, we analyze the confidence of the decision tree policy in its decision. To do this, we look at the difference between the maximum and minimum Q values for the same observation as perceived by the RL policy [19] (lines 11-13). If the difference is below a defined threshold $\beta$, it can have two meanings. 1) Any action could be considered as a near-optimal action for the given observation or, 2) the RL policy is not confident about which action can produce better outcomes in the future. Although we are only concerned about the latter, it is safe to classify the observation as a blind spot. This condition captures learning limitations faced by the RL policy and thus the decision tree policy. Such limitations can occur because of issues in state representations, credit assignment, and architectural and data limitations, etc. (Figure 2d in our example).

*c) Imitation consistency:* Finally, due to its relatively limited capacity, the decision tree policy may not perfectly imitate the RL policy (Figure 2e in our example), and this must be flagged as a blind spot (lines 15-16).

### *E. Offline Supervision*

Offline supervision refers to the asynchronous supervision performed by traffic signal control engineers manually at control centers. With the online supervision in place, offline supervision can be performed periodically to review and incorporate the online supervision suggestions permanently into the control policy. Such updates to the control policies require manual human interventions to vet the quality of the overridden actions. Another practical use case of offline supervision is to respond to community-raised issues. Changes to the control policies are needed to be made as necessary in response to the issues raised. We propose an interactive dashboard *DTLight*, which offers one viable pathway to facilitate offline supervision of decision tree policies.

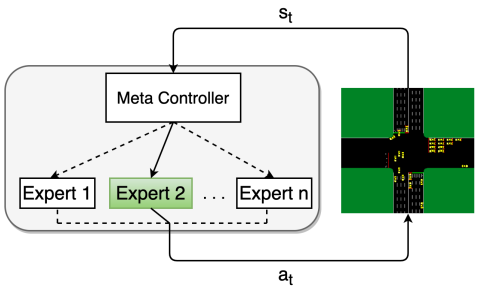



Fig. 3: Schematic view of *DTLight* architecture

Inspired by Mixture of Experts (MoE) architecture [21], we design the architecture of *DTLight* such that each traffic signal (intersection) is controlled by a meta controller and a set of expert policies. Each expert is specialized in a defined part of the state space. Given an observation, the meta controller picks the expert that will be used to obtain the control action. We represent all experts and the meta controller as decision tree policies. One expert will always be the decision tree devised using the RL policy (main expert). The engineers would gradually create new experts in response to online

supervision logs and community-raised issues. To facilitate that, *DTLight* provides user-friendly interfaces for creating and modifying expert decision tree policies and reviewing and permanently accommodating online supervision logs. The schematic overview of the architecture is illustrated in Figure 3. An example instance of the meta controller interface of *DTLight* is shown in Figure 4.

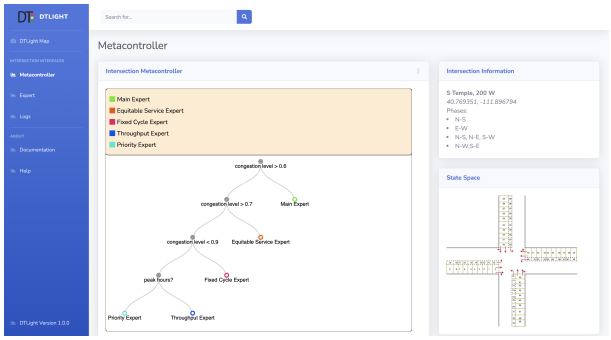



Fig. 4: *DTLight* meta controller interface

## V. EVALUATION

In this section, we experimentally validate our method. We first show that controlling traffic signals using decision tree policies devised based on expert RL policies are just as performant as the experts. Next, we show that the proposed MAS framework yields overall performance improvement on a scenario derived from a real traffic situation.

### A. Evaluation of decision tree policies

We first evaluate the performance of devised decision tree policies. We use the Max-Pressure [8] method as a baseline control policy and DQN [18] for training the expert agent. We use the Viper algorithm (Section III-B) to devise decision tree policies using the CART algorithm [22]. We model a 4-way intersection with left and right turns and four traffic phases. Each approaching and outgoing road has four lanes. The four phases include two straight-going phases and two left-turn phases. A Weibull distribution of shape two was used to generate vehicle arrivals. Each vehicle has a 25% chance to turn left or right and a 75% chance to go straight. The RL policy used as the expert has an input layer of 80 neurons, five hidden layers of 400 neurons each, and an output layer of 4 neurons representing the four phases as actions. All simulations are run on SUMO[1] traffic simulator.

In Figure 5a, we illustrate the performance of devised decision tree policies compared to the expert RL policy and the baseline Max-Pressure policy. In general, we would expect the decision tree policies to underperform slightly than the expert model used to train them because of the limited learning capacities of the decision tree policies. Surprisingly, in our experiments, decision trees with depths 7 and 8 slightly outperform the expert RL policy used to train them. We believe this happens because of the better generalization capability of decision tree policies due to their under-parameterized nature. In Figure 5b, we show the change of average waiting time of the approaching vehicles with increasing tree depth. Our decision trees policies with depths 7 and 8 are just as performant as the RL policy. In general, we prefer shallower decision trees as they are easier to interpret. Therefore, for the rest of the experiments, we choose the decision tree policy with depth 7.

[1]https://www.eclipse.org/sumo/

### B. Evaluation of online supervision and transfer of control

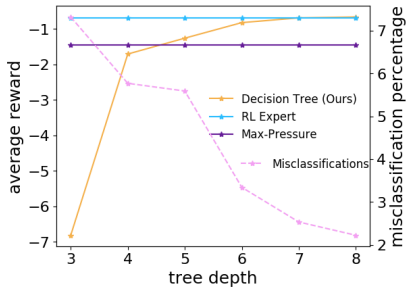
Next, we show the benefit of online supervision and transfer of control in scenarios derived from real-world traffic situations, which may exhibit out-of-distribution environment dynamics. Specifically, during evaluation time, we consider a scenario in which emergency vehicles should be given the right of way. Therefore, the traffic control policy should clear the traffic to facilitate the emergency vehicle to pass through the intersection with minimum wait. We modify the Max-Pressure control policy to accommodate this requirement and use it as the online fail-safe policy. The training simulations do not specifically model emergency vehicles, and therefore the trained RL policy has not learned to treat emergency vehicles differently. We define real-world traffic signal control objective as $\min\{w_{emergency} + w_{non-emergency}\}$ where $w_{emergency}$ and $w_{non-emergency}$ are the waiting times of emergency and non-emergency vehicles, respectively.

We present the performance of the online supervised policy (Definition IV.6) in comparison to baseline policy in Figure 5. Figure 5c shows the total waiting time as defined by the objective above. Our proposed online supervised policy achieves overall low waiting times by transferring the control from decision tree policy to fail-safe policy as per Algorithm 2, shown in Figure 5f. It is also able to outperform the performance of carefully defined Max-Pressure policy in terms of waiting time for emergency vehicles as shown in Figure 5d. Recall that both the RL policy and the decision tree policy were optimized to minimize the non-emergency vehicle waiting times and not on the overall objective; thus they have significantly higher waiting times compared to the supervised policy. Conversely, our supervised policy has comparatively high waiting times for non-emergency vehicles compared to the decision tree policy and the RL policy (Figure 5e). In summary, our proposed supervised policy achieves the best of both worlds between the safety of the fail-safe policy and the performance of the RL policy.

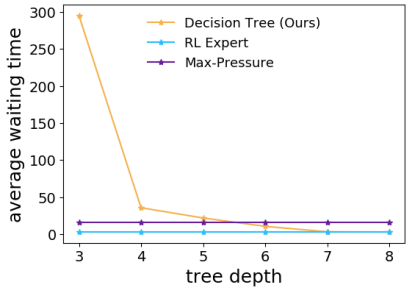
## VI. CONCLUSION

In this work, we investigate the use of decision tree policies trained to imitate reinforcement learning policies for traffic signal control. We propose Mixed Autonomous Supervision (MAS) and a two-part supervision model with online automated supervision and offline human supervision to implement it. Our results show those decision tree policies devised based on expert RL policies are just as performant as the experts and that the proposed MAS framework yields overall performance improvement.

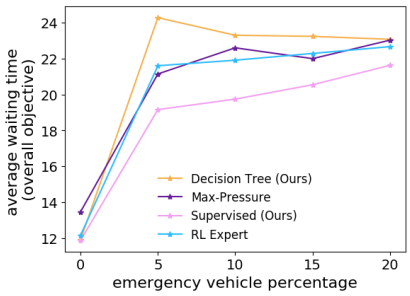
Future directions of research include extending the blind spot detection algorithm to relax the assumptions made, be



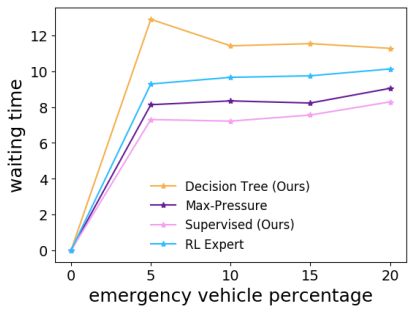

(a) Average reward and misclassification percentage with increasing tree depth



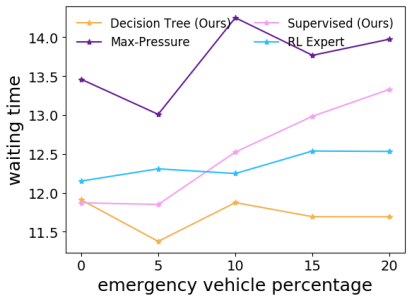

(b) Average waiting time versus tree depth



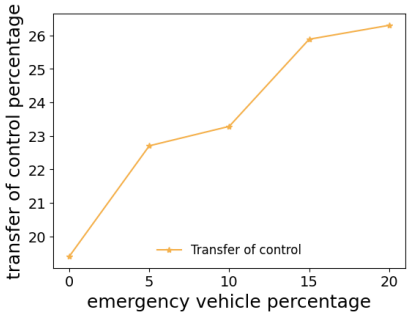

(c) Objective: average waiting time of emergency and non-emergency vehicles




(d) Average waiting time of emergency vehicles




(e) Average waiting time of non-emergency vehicles




(f) Online supervision: transfer of control percentage

Fig. 5: Performance of the approach with varying tree depths and emergency vehicle percentages.

rigorous in identifying out-of-distribution blind spots and accommodate continuous actions. Additionally, extending the proposed method to coordinated traffic signal control is also a possible future work. Finally, we plan to investigate devising shallower decision trees by leveraging the symmetries present in the intersection designs.

## VII. Acknowledgement

The authors would like to acknowledge funding support from the MIT-IBM Watson AI Lab for this work. The authors would also like to thank the Utah Department of Transportation for informative discussions and feedback.

## References

[1] Kok-Lim Alvin Yau, Junaid Qadir, Hooi Ling Khoo, Mee Hong Ling, and Peter Komisarczuk. A survey on reinforcement learning models and algorithms for traffic signal control. 2017.

[2] David Levinson. Speed and Delay on Signalized Arterials. Technical report, 1998.

[3] P. Lowrie. Scats: Sydney co-ordinated adaptive traffic system: a traffic responsive method of controlling urban traffic. 1990.

[4] P. Hunt, D. Robertson, R. Bretherton, and M. Royle. The scoot on-line traffic signal optimisation technique. *Traffic engineering and control*, 23, 1982.

[5] Stephen James et al. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[6] Michael Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. Springer International Publishing, Cham, 2017.

[7] F. Webster. Traffic signal settings. Technical report, Road Research Technical Paper No 39, Road Research Laboratory, 1958.

[8] Pravin Varaiya. *The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections*. Springer New York, 2013.

[9] Carlos Gershenson. Self-organizing traffic lights. 2005.

[10] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. KDD '18, page 2496–2505, 2018.

[11] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. 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*, 2019.

[12] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, Nov 2019.

[13] James Ault, Josiah P. Hanna, and Guni Sharon. Learning an interpretable traffic signal control policy, 2020.

[14] Ramya Ramakrishnan, Ece Kamar, Besmira Nushi, Debadeepta Dey, Julie Shah, and Eric Horvitz. Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), Jul. 2019.

[15] Kyle Hollins Wray, Luis Pineda, and Shlomo Zilberstein. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 517–523. AAAI Press, 2016.

[16] Federal Highway Administration. Automated traffic signal performance measures.

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[19] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in Neural Information Processing Systems*, pages 2494–2504, 2018.

[20] Cathy Wu. *Learning and Optimization for Mixed Autonomy Systems: A Mobility Context*. PhD thesis, 2018.

[21] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 1991.

[22] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 2011.