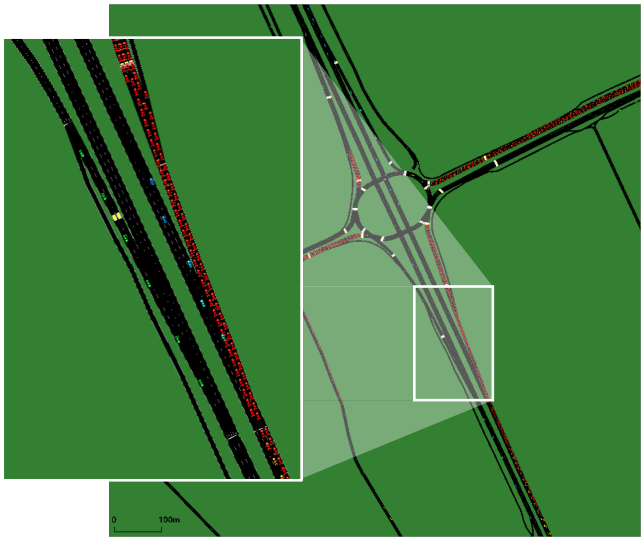

**Figure 1: A long queue blocking highway traffic flow, despite the highway being grade-separated from the roundabout.**

The contributions of the paper are manifold:

(1) A Reinforcement Deep Learning method for signalized roundabouts in congested network, a complex traffic light control scenario that, to the best of our knowledge, has not been addressed by previous works. Previous studies featured simple two or four-way intersections.
(2) An episodic formulation of the traffic light control problem for traffic jam avoidance and faster convergence.
(3) A policy gradient method with a time baseline, to reduce the variance in episodic settings, where the accumulated reward at each time step $t$ is monotonically decreasing with respect to $t$.
(4) An experimental evaluation on a real roundabout, simulated using traffic data recorded in the peak hour of a weekday.

The experimental results show that our method reaches higher performances in several measures including average speed, halting times and emissions, while being able to avoid the overflowing high volume traffic to interfere with other junctions.

The rest of the paper is structured as follows. Section 2 describes variables and dynamics involved in designed a signal timing policy. Section 3 gives a brief introduction to the RL technique of Policy Gradient and formulate the problem as a RL task. Section 4 presents the proposed method based on episodic conditions and Policy Gradient with time baseline. Section 5 describes the experimental setup and shows the results. Section 6 provides related work in the area of traffic light control with a focus on RL techniques and Section 7 concludes.

## 2 TRAFFIC MODELING

We start by deriving the basic equation, from first principles, of traffic signal control. The result is very intuitive, namely that to minimize overall delay, phase duration should be proportional to the arrival rate. However, as we will see the derivation depends on simplistic assumptions. When these unrealistic assumptions are relaxed, more complex machinery is required and we claim that RL methods are suitable for designing traffic signal timing policies in realistic settings.

Consider a simple scenario where a North-South (NS) and an East-West roads meet at an intersection. Suppose the traffic (cars) follow a poisson process and starts arriving from time $t = 0$ and the total traffic cycle time is $T$. Let $\lambda_1$ and $\lambda_2$ be the rate of traffic arrival on the NS and EW roads respectively. Then by definition of the poisson process the expected number of cars at time $t$ is $\lambda t$ and they are uniformly distributed in the interval $(0, t)$. Assume NS gets the green signal first during which time the EW has the red signal. Let the duration of the green phase be $\alpha$. Therefore the duration of the red phase is $(T - \alpha)$. Cars arrive uniformly. The average delay on NS is $\frac{T-\alpha}{2}$ and on EW is $\frac{\alpha}{2}$. Thus the total delay, $D(\alpha)$ is

$$D(\alpha) = \frac{\lambda_1 (T-\alpha)^2}{2} + \frac{\lambda_2 \alpha^2}{2}$$

To minimize the total delay set $\frac{\partial D(\alpha)}{\partial \alpha} = 0$ and solve for $\alpha$. Thus

$$-\lambda_1 (T-\alpha) + \lambda_2 \alpha = 0 \tag{1}$$

$$\alpha = \frac{\lambda_1}{\lambda_1 + \lambda_2} T \tag{2}$$

We generalize the above green splitting equation to consider the case of $m$ exclusive flows, each with a rate $\lambda_i$, thus the duration is

$$\alpha_i = \frac{\lambda_i}{\sum_{i=1}^{m} \lambda_i} T \tag{3}$$

### 2.1 Additional delay components

There are several variables that should be taken into account in order to accurately model the delay and consequently optimize the green splitting. We replace the traffic cycle time $T$ with the effective green time $t_g$, as the total effective green time available in a cycle. The effective green time is defined as

$$T_g = T - n(d_a + d_q + d_c) \tag{4}$$

That is, the total time $T$ minus a *lost time* for each phase as the sum of three delays:

- Approaching delay $d_a$ is the time lost for decelerating before entering the queue.
- Queue delay $d_q$ is the period of time between entering the queue and reaching the STOP line.
- Control delay $d_c$ is the time lost for discharging the central area (all red phase) before changing phase, this includes the startup time where acceleration is considered.

With few assumptions on vehicles features, the approaching and control delays $d_a$ and the control delay $d_c$ are constant. Conversely, it is hard to formulate an accurate queue delay model. The queue delay $d_q$ is a non-linear function on the number of cars and it should model acceleration, stop and go waves phenomena, and it may span across multiple cycles.

### 2.2 Traffic model in signalized roundabout

Complex junctions such as signalized roundabouts present additional features and issues that are not addressed in traditional transportation models.

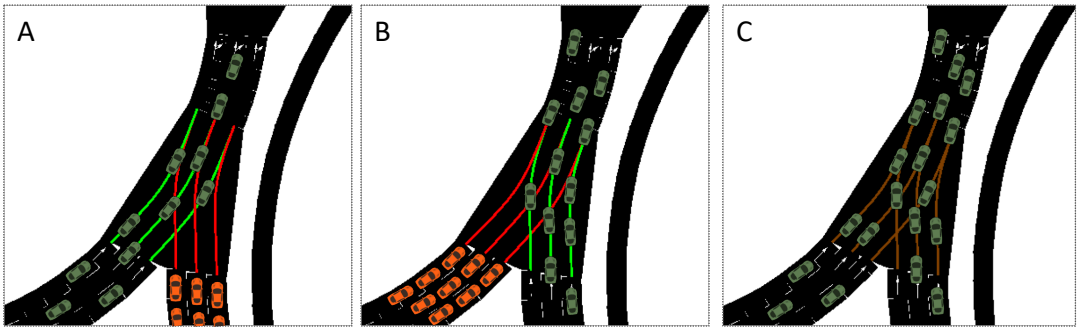


**Figure 2: Phases of a weaving section in the signalized roundabout subject of the study. In phase A, incoming flow is halted and the circulatory lanes are in free flow, allowing the roundabout to be emptied. In phase B, the vehicles are halted inside the roundabout, incoming flow is allowed to enter and exit (only U-turn exit will require them to wait). In phase C, incoming flows yields and the two conflicting flows weave autonomously.**

*2.2.1 Roundabout flows weaving.* In signalized roundabouts, the circulatory lanes are the shared resource for which approaching arms are competing. However, the way they are accessed and occupied is different from the central area of an intersection:

- In contrast to intersections, roundabouts have one-way traffic flows. Vehicles from different entry arms can access the shared resource in a safe way by yielding (Figure 2-C).
- The period of time a vehicle spends in the shared resource is longer on average even when there is no congestion. Consider for example the time needed to reach the last exit of a large roundabout in comparison with passing an intersection.
- Vehicles can be halted at a traffic light inside the shared resource but still allowing an incoming flow to enter and leave the roundabout meanwhile (Figure 2-B)

While it is possible to wait for the circulatory lanes to be completely emptied before moving to the next green phase, this would be impractical and it will lose the advantages of having a roundabout. On the other hand allowing the two flows to weave, as in a non-signalized intersection, makes the startup delay less predictable for two reasons: (i) incoming vehicles have to yield before entering the roundabout and (ii) once in the roundabout, the vehicles may have to change lane depending on the exit they will take.

In flows weaving, the entry capacity of a roundabout arm must take into account the circulating flow: entry capacity decreases as the circulating flow increases and vice versa.

## 2.3 O-D driven green splitting

The destination of vehicles is another variable that an optimal policy should be taken into account in choosing whether to alternate or weave-in the incoming traffic flows. Depending on the destination of vehicles, it is possible that traffic flowing from different incoming arms does not compete for the same circulatory lanes. This scenario would allow for multiple incoming lanes to have access to the roundabout at the same time, without any conflict or weaving. For example, given that all the vehicles coming from the South will take the north exit and vice versa, both incoming lanes from north and south may have the green light at the same time without interfering (see Figure 3 for other examples).

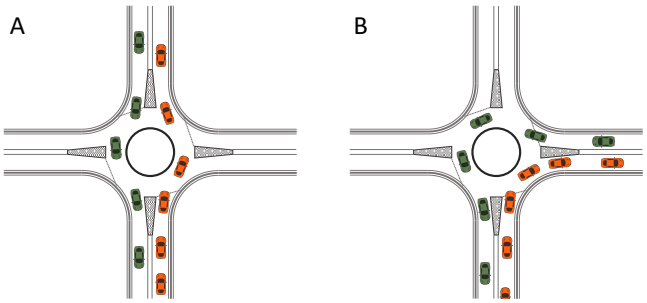


**Figure 3: Examples of non-competing routes flows: (A) NS-SN, (B) SE-ES, (C)**

While modeling an origin-destination (O-D) driven phasing without knowing the destination of vehicles is not possible, a fair assumption can be made on the U-turn traffic (e.g. from north to north) to be lower then other routes.

In Table 1 we assume, given an entry arm in a four-way roundabout, a probability of 0.1 for the U-turn and uniform distribution of 0.3 for each remaining routes. The matrix in Table 1 shows the probability of a vehicle coming from an incoming arm $X_i$ to occupy the circulatory lane of the weaving section of $Y_c$.

With the U-turn assumption, the pair of incoming arms (N,S) and (E,W) have less probability of conflicts. Modeling an O-D driven phasing does still need the knowledge of routes probability, but the additional phases of less conflicting routes can be explored in a Reinforcement Learning setting to learn a complex policy.

# 3 PROBLEM FORMULATION

The problem of finding an optimal strategy for traffic light phase selection depending on the vehicular network state can be framed as a reinforcement learning task. After introducing some background and notation, we model the traffic light control problem in the reinforcement learning framework.

| Occupancy probability | | | | |
|---|---|---|---|---|
| | Circulatory lanes | | | |
| Entry arms | $N_c$ | $W_c$ | $S_c$ | $E_c$ |
| $N_e$ | 1 | 0.7 | 0.4 | 0.1 |
| $W_e$ | 0.1 | 1 | 0.7 | 0.4 |
| $S_e$ | 0.4 | 0.1 | 1 | 0.7 |
| $E_e$ | 0.7 | 0.4 | 0.1 | 1 |

**Table 1: Occupancy probability of each circulatory lane (or group of lanes) by incoming traffic from each entry arm, based on the assumption of a uniform O-D matrix and 0.1 probability of U-turn.**

## 3.1 Reinforcement learning

Reinforcement learning is a goal-directed learning task, in which an agent interacts with the environment in order to learn an optimal action selection policy [22].

At each discrete time step $t = 0, 1, 2, 3, \ldots$, the agent receives the current state $S_t \in \mathcal{S}$ from the environment, and select an action $A_t \in \mathcal{A}(s)$ according to its policy. Together with the next state $S_{t+1}$, the agent receives also a *reward* $S_{t+1}$, that is, a scalar signal that we want to maximize. More specifically, assuming that the number of time steps $T$ is finite within one *episode*, the goal of optimal action selection at timestep $t$ is to maximize the cumulative discounted reward $G_T$:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-1} R_T, \tag{5}$$

where $\gamma$ is a *discount rate* parameter, with $0 \leq \gamma \leq 1$, that determines the present value of future rewards.

The model of the environment is defined as a Markov decision process (MDP), where the probability distribution for the random variables $R_t$ and $S_t$ is defined by the p function

$$p(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}, \tag{6}$$

that describes the dynamics of the MDP. The typical reinforcement learning problem is model-free, meaning that the MDP and its associated $p$ function are not known. Instead, exploratory actions (i.e., non optimal) are needed in order to estimate the model dynamics through a large number of interactions.

The ultimate goal of reinforcement learning is to learn a policy $\pi_*(a|s)$, that is, a mapping from states to probabilities of selecting each action, that maximize the final reward. This optimal policy $\pi_*$ can be obtained by either estimating the value of actions through **action-value methods**, or by directly learning a parameterized policy through **policy gradient methods**.

Action-value methods learn the optimal value of actions for each state $q_*(s, a)$, which is defined through the Bellman optimality equation [22]:

$$q_*(s, a) = \sum_{s', r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]. \tag{7}$$

In policy gradient methods estimating the action value or the state value is not required, instead a parameterized policy $\pi(a|s, \theta)$ is learned in order to maximize some performance measure $J(\theta)$ based on the accumulated reward. The optimal policy in these methods is approximated by the iterative update

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t). \tag{8}$$

The policy $\pi(a|s, \theta)$ can be any parameterized function as long as the partial derivatives with respect to parameter vector $\theta$ exist and are finite for all the states and actions. In practice, a neural network is used and an exponential softmax is applied on its output nodes in order to obtain a probability distribution from the numerical preferences.

## 3.2 Traffic light control environment

We represent the signalized vehicular network environment as a MDP with states, actions and rewards. We design the MDP so that states and reward can be easily measured in a real scenario (e.g. based on loop detectors instead of aerial images and travel times) and the actions to be realistic and safe.

*3.2.1 States.* The state in our environment is defined as the set of car counts coming from the detection loops installed on the network. In a roundabout network, detectors loops are installed on both approaching and circulatory lanes. For approaching lanes, two lines of detectors are installed, one line closer to the roundabout entrance, one several meters farther (e.g. 150m before the entry). A line of detectors has one detector for each lane.

Let $D_i(t)$ the number of cars that have been over the detector $i$ during a time step $t$ we define the state as $s_t = (D_1, D_2, \ldots, D_m)$ with $m$ the number of detectors.

*3.2.2 Actions.* The actions allow to control the flow by changing the phase of the traffic lights. In the proposed environment, instead of setting a binary action to move to the next phase, the agent can select any phase at each time step without particular order. We avoid employing a binary action of phase switching – that is, to move to the next phase in the cycle – as the policy would have no information about the current or next phase. It has also been shown that including the current phase in the state it is not sufficient in the setting of phase switching action [26], and a separate set of neural network weights would be needed for each phase.

| Action | Entry lanes | Circulatory lanes |
|---|---|---|
| $a_1$ | $G(N_e); R(W_e, S_e, E_e)$ | $G(W_c, S_c, E_c); R(N_c)$ |
| $a_2$ | $G(W_e); R(S_e, E_e, N_e)$ | $G(S_c, E_c, N_c); R(W_c)$ |
| $a_3$ | $G(S_e); R(E_e, N_e, W_e)$ | $G(E_c, N_c, W_c); R(S_c)$ |
| $a_4$ | $G(E_e); R(N_e, W_e, S_e)$ | $G(N_c, W_c, S_c); R(N_c)$ |
| $a_5$ | $G(N_e, S_e); R(W_e, E_e)$ | $G(W_c, E_c); R(N_c, S_c)$ |
| $a_6$ | $G(W_e, E_e); R(N_e, S_e)$ | $G(N_c, S_c); R(W_c, E_c)$ |
| $a_7$ | $G(N_e), A(S_e); R(W_e, E_e)$ | $G(W_c, E_c); A(S_c); R(N_c)$ |
| $a_8$ | $G(S_e), A(N_e); R(W_e, E_e)$ | $G(W_c, E_c); A(N_c); R(S_c)$ |
| $a_9$ | $G(W_e), A(E_e); R(N_e, S_e)$ | $G(N_c, S_c); A(E_c); R(W_c)$ |
| $a_{10}$ | $G(E_e), A(W_e); R(N_e, S_e)$ | $G(N_c, S_c); A(W_c); R(E_c)$ |
| $a_{11}$ | $R(N_e, W_e, S_e, E_e)$ | $G(N_c, W_c, S_c, E_c)$ |

**Table 2: Phases actions for a four-way roundabout.**

We provide one phase for each set of non-conflicting flows, and one phase for each pair of nonconsecutive approaching lanes, assuming that the U-turn is a less frequent route, where two lanes

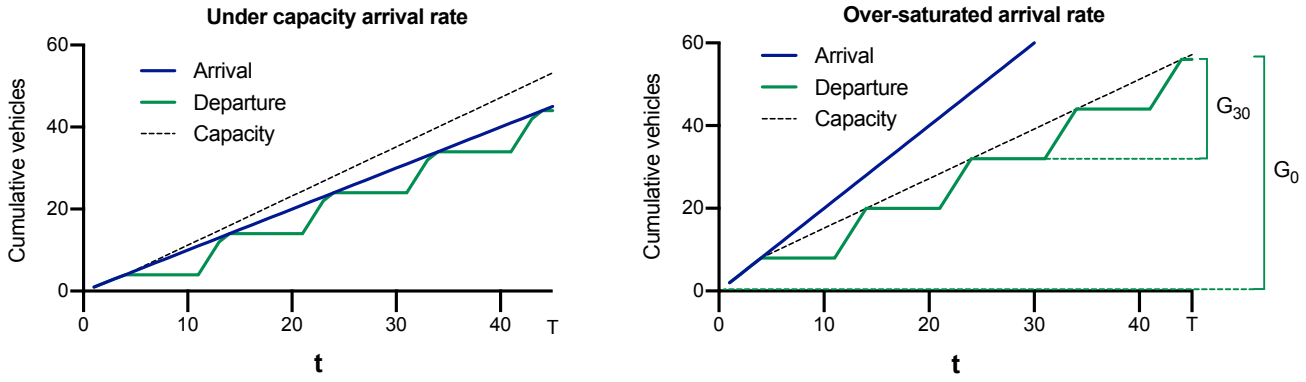


**Figure 4: Cumulative vehicles illustration of over-saturated (left side) and under capacity arrival rate for the same intersection. In condition of saturation the capacity function can be estimated from the departure function.**

have both the green light and all other traffic lights are red. We also provide, for the previous pairs, the combinations of phases for which one entry arm of the pair has green and the other has an intermittent amber (i.e. approaching cars must yield). Finally, an additional *all-red* phase is provided to allow the discharge of the circulatory lane when congested.

In Table 2 we list the set of 11 actions, each one corresponding to a traffic light phase, for a four-way roundabout. In order to safely switching between any pair of phases, we create an amber light phase transition, hidden from the agent, for each possible pair of distinct phases. Conversely, no actual change is made on the phases when the action selected is equal to the previous.

*3.2.3 Reward.* As shown in Section 2, the goal of traffic light optimization in a saturated network has a dual objective: maximizing the capacity of the intersection and avoid bad traffic conditions such as long queues that may interfere with other junctions. The reason why these two objectives are competing is that by giving a long green time to one of the saturated entry it is possible to reduce lost time and obtain very high capacity, at the detriment of causing a traffic jam elsewhere. We trade-off between these two objectives by considering the capacity in the reward function, while avoiding traffic jams using episodic conditions (see Section 4).

We have also shown that deriving an analytical a traffic model to measure the capacity of a signalized roundabout with a given timing is extremely difficult, because of the many dynamics involved such as flows weaving, O-D dependent occupancy and combined phases.

However, since we are merely interested in the current performance of our policy, it is possible to measure empirically the capacity as the vehicle departed over a period of time. In Figure 4 we show the cumulative arrival function (vehicles arriving piling up in front of at the roundabout) and departure function (vehicles leaving the roundabout). When the roundabout is saturated, the capacity is proportional to the number of departed vehicles. On the contrary, when the arrival rate is lower than the capacity, it is not possible to estimate the capacity, as the arrival rate becomes the upper bound of the departure rate. Since our simulations are always over-saturating the roundabout by design, we can assume that the number of vehicles left the roundabout from time $t$ to time $T$, $L(T) - L(t)$ is a valid approximation of $C(T) - C(t)$, where $C(t)$ is the cumulative capacity function at time $t$, that is $C(T) \approx L(T)$.

We define the reward function, $R(t) = C(t) - C(t-1)$ as the number of vehicles departed from the roundabout at time $t$. It follows that the return at time $t$ is $G_t = L(T) - L(t)$ and the total return of an episode is $G = G_0 \approx C(T)$. In the over-saturated plot of Figure 4 we show as examples $G_{30}$ and $G_0$ using Equation 5 and the defined reward.

# 4 METHOD

## 4.1 Episodic Conditions

Episodic conditions are boolean conditions verified after each step of an episode to determine if the *terminal state* of an episode, $S_T$, is reached. In a non-discounted setting, the return $G_t$ is defined as the sum of the rewards $R_{t+1} + R_{t+2} + \ldots + R_T$, with T being the time of termination. Thus, the expected return increases with T and each episode is a repeated attempt of avoiding the episodic condition.

We design traffic jam related episodic conditions so that the optimal policy is forced to avoid them in order to maximize the accumulated reward. A secondary effect is that enforcing the policy to avoid traffic jams will lead to faster convergence in Montecarlo setting, as it will generate shorter bad episodes. Conditions we want to avoid are:

(1) **Long blocking queues.** When a long queue piles up in front of a traffic light, this has negative consequences even after the green light is given, because of the stop-and-go waves. More importantly, a long queue has negative side effects on other junctions. As an example, Figure 1 shows a real scenario in which the queue from the entry arm of an intersection is blocking the traffic flow of the highway. In the example, the roundabout interchange is designed so that the highway and the roundabout are on two different heights, however, due to a suboptimal traffic light control policy the roundabout traffic is interfering with the highway.

(2) **Long waiting times.** In a very common scenario the flows from different entry arms are unbalanced, however the rewarded measures from each entry arm are usually averaged or summed [16, 26]. This means that, for example, when there is only one vehicle for one direction and heavy flow in the other, the vehicle may have to wait a long time before the agent switches phase.
(3) **Zero car exiting.** A particularly poor action selection occurs when the green phase is given to an empty entry arm, a condition known as *green-idling*. This results in no vehicles exiting the intersection or the roundabout for a whole time step even if other vehicles are halted waiting. This condition will trigger when there is a traffic jam inside a roundabout or an intersection (i.e. *cross-blocking*).

Despite an optimal policy from a well-designed reward should inherently avoid the above conditions, there would be no strong enforcement for these. Moreover, taking into account all the above conditions inside a reward function would require a combination of several variables such as max and average waiting time, max and average queue length. Some of these variables may require complex detectors (plate numbers tracking) while their combination would need a search for the right hyper-parameters, to weight each one of them.

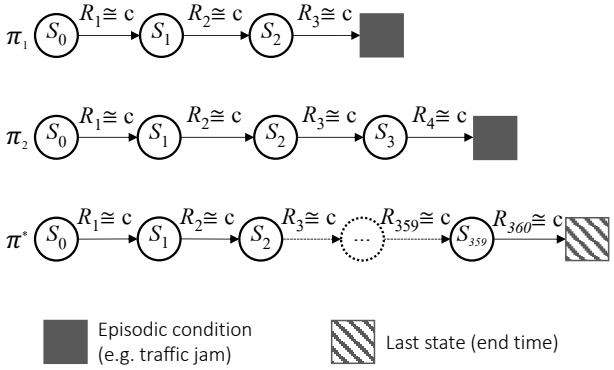


**Figure 5: The optimal policy $\pi*$ accumulates reward $R_i$, approximately equal to the instant capacity $c$, for each time step period until the maximum length of the episode. Suboptimal policies ends by episodic conditions such as traffic jams.**

## 4.2 Time Baseline

A crucial aspect of Policy Gradient methods is dealing with the variance of the gradients in different states and episodes [22]. In the policy gradient theorem, $\nabla J(\theta) \propto \mathbb{E}[(q(s,a))\nabla\pi(a|s,\theta)]$, the gradient is proportional to the discounted return from the state $s$. However, the range of this return is heavily environment-dependent, so that one state may have a very low return while another state a very high one. This can negatively affect the training dynamic, therefore it is very common to subtract a *baseline* from the accumulated reward in order to normalize the gradient scale in the optimization step. For example, one choice for the baseline can be the average of the accumulated reward obtained so far. In this case, the REINFORCE algorithm would positively update the parameters only if the accumulated reward has been greater than the average. However, some states inherently bring smaller or higher rewards, independently from the action taken, therefore the baseline should change accordingly. For this reason, the baseline is usually chosen as a function of the state, as in the Actor-Critic methods.

In the episodic case, the number of remaining steps until the final state $T$ can also be an important source of variance. In the classical REINFORCE algorithm [22], the parameter increment is proportional to the accumulated reward $G_t = \sum_{k=t+1}^{T} R_k$, that is, the complete return from time step $t$ to the final state $T$. In several RL setting such as the CartPole environment [3, 20], in which a positive reward is accumulated at each step, the longer the episode lasts the greater is the return. Conversely, in a single episode, the closer a state is to $T$ the lower is its return.

This variance cannot be captured merely by a function of the state. Consider for instance an environment with only positive rewards. A state $S$ could appear in two different time steps $t$ and $t+k$, with $S_t = S_{t+k}$ thus $v(S_t) = v(S_{t+k})$, while the relative returns would be $G_t \geq G_{t+k}$ by definition of $G_t$. On the other side, adding the time information in the state is conceptually wrong, as the action selection should only depend on the current traffic situation (e.g. the state of the detectors or the number of cars for each lane). It is therefore reasonable that only the baseline should vary with the time step.

We denote by $B_t$ a return baseline at time $t$, that can be computed as the average, or as the weighted moving average, of discounted return at time $t$, $G_t$, in the generated episodes. The resulting policy gradient theorem becomes

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - B_t)\nabla\pi(a|s,\theta) \tag{9}$$

$$= \mathbb{E}_\pi \left[ \sum_a q_\pi(S_t,a)\nabla\pi(a|S_t,\theta) \right] \tag{10}$$

where $\mu(s)$ is the on-policy distribution under $\pi$. The equation still holds because the subtracted quantity is zero, as the temporal baseline does not vary with the selected action:

$$\sum_a B_t \nabla\pi(a|s,\theta) = B_t \nabla \sum_a \pi(a|s,\theta) = B_t \nabla 1 = 0 \tag{11}$$

Therefore, the update rule of the REINFORCE using the time baseline becomes

$$\theta_{t+1} = \theta_t + \alpha(G_t - B_t)\nabla \ln \pi(A_t|S_t,\theta_t) \tag{12}$$

In Algorithm 1 we define the Monte-Carlo Policy Gradient algorithm with time baseline using the above defined update rule and an exponentially weighted moving average (EWMA) for updating the baseline $B_t$ in every episode. Input hyper-parameters the learning rate $\alpha$ and $\beta$ the degree of weighting decrease to compute the $B_t$ moving average. By setting $\beta = \frac{1}{t}$ the mean of $G_t$ will be used instead.

**Algorithm 1** Montecarlo REINFORCE with EWMA time baseline

**Input:**

(1) Differentiable policy parametrization $\pi(a|s,\theta)$
(2) Learning rate: $\alpha > 0$
(3) Discount factor: $\gamma > 0$
(4) Degree of weighting decrease for $B_t$: $\beta > 0$
(5) Maximum episode length $m$

Initialize $\theta \in \mathbb{R}^d$ (e.g. to $\overrightarrow{0}$)
Initialize $i = 0$
Initialize $B \in \mathbb{R}^m$ to $\overrightarrow{0}$
**for each** episode **do**
  Generate episode $S_0, A_0, R_1, ..., S_T, A_T, R_T$ using $\pi_\theta$
  **for each** t in [0 ...T] **do**
    $G_t \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\delta \leftarrow G_t - B_t$
    $\theta \leftarrow \theta + \alpha\gamma^t \delta \nabla \ln \pi(A_t|S_t,\theta)$
    **if** $i = 0$ **then**
      $B_t \leftarrow G_t$
    **else**
      $B_t \leftarrow \beta G_t + (1-\beta)B_t$
    **end if**
  **end for**
  $i \leftarrow i + 1$
**end for**

Note that in our setting of heavily congested traffic and capacity-based reward, the state itself is not crucial in the computation of the baseline, because the capacity is saturated thus any state of the detectors, at a given time $t$, will lead to a similar final return. Conversely, a temporal baseline is needed to converge faster to an optimal solution. Moreover, since the policy is still stationary and it does not depend on the time, the action selection is only based on the detectors state.

### 4.3 Generalization and exploration

RL agents are prone to overfit on the environment seen in training time while failing when first encountering a new environment for which the distribution of the states is different [5]. In order to allow the training from different distributions of states, we implemented a procedural generation of environments with the same traffic volume but different direction rates as shown in Figure 6.

Another requirement to find an optimal policy is ensuring that, after several episodes, the agent is still exploring new trajectories from time to time. Ensuring that $\pi(a|s) \in (0,1)$ for all $s, a$ is usually not enough, we need instead a form of regularization such as entropy regularization [17]. Entropy $\mathbb{H}(s,\pi)$ of the actions probability distribution in policy $\pi$ for state $s$ is defined as

$$\mathbb{H}(a,\pi) = -\sum_a \pi(a|s) \log \pi(a|s) \tag{13}$$

Entropy is minimum when $\pi(a|s) = 1$ for one action and 0 for all the others, that is, when it becomes a deterministic policy. Since we want the entropy to be high, we subtract the entropy from the loss before the optimization step of the neural network.

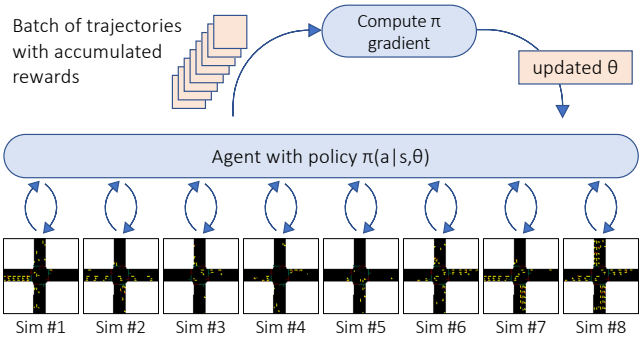


**Figure 6: Parallel experiences from 8 simulations with procedurally-generated traffic flows.**

## 5 EXPERIMENTS

**Setup.** We run all the evaluation experiments on a real roundabout dataset. The exact network shape includes an area of 2.4km × 3.4km, which surrounds the Al Gharrafa roundabout in Doha, Qatar, as shown in Figure 9. For these experiments the simulated traffic data has been recorded during the peak hour of a weekday and reproduced in the simulator following the real origin-destination matrix.

Training and evaluation are carried out using the SUMO simulation software (Simulation of Urban MObility)[1]. We implemented an Open AI gym module[2] to create the simulation environment for the roundabout. The environment has several versions – to enable/disable the GUI visualization, enable/disable the episodic conditions) or to run one of non-RL policy.

The policy function is implemented as a fully connected neural network with 68 input nodes, a hidden layer of 512 nodes with ReLu activation and we apply a softmax to the 11 output nodes, each one representing the probability of one action.

**Results.** The overall results averaged over 10 simulations of one hour of recorded real traffic are shown in Table 3. In each simulation we measure the total accumulated reward (Return column), the percentage of time in the episode in which the traffic approaching the roundabout was blocking other junctions, the total number of vehicles that completed their trip in the whole network, the average speed of vehicles, the average of halted cars at each time step, the average spent by each vehicle waiting, the sum of emissions of all cars. Except for the number of completed trips, all the measures are taken from the edges approaching the roundabout or inside the roundabout. Note that in the evaluation, we do not enforce episodic condition in order to take measurements for the full hour.

Non-reactive policies based on the rate of each entry (i.e. Rate aware in Table 3) use the double entry phases (actions $a_5$ and $a_5$) with phase duration proportional to the arrival rate, as described in Section 2. We provide results for two common cycle lengths of 120 and 170 seconds. Both policies produce very high waiting times due to prolonged congestions that last half of each episode, it is clear however the advantage of having longer phases. Our PG method with EWMA time baseline and $\beta_t = 0.5$ achieves the lowest rate

[1] https://sumo.dlr.de/
[2] Publicly available at https://github.com/qcri/gym-gharrafa

| Policy | Return | % Blocking | Complete trips | Mean speed [m/s] | Mean halted cars | Waiting time [s] | $CO_2$ [kg] |
|---|---|---|---|---|---|---|---|
| Rate Aware Short | 1598 | 68.1% | 4973 | 7.48 | 333 | 2283 | 7675 |
| Rate Aware Long | 2076 | 49.4% | 6224 | 7.45 | 226 | 2202 | 5763 |
| PG Mean Baseline | 3036 | 12.1% | 6550 | 9.60 | 87 | 682 | 2942 |
| PG Time Mean | 2694 | 6.7% | 6913 | 9.62 | 69 | 171 | 2681 |
| PG Time EWMA | 2956 | 6.4% | 6847 | 10.01 | 52 | 116 | 2178 |

**Table 3: Performances for traffic models policies, classic PG policies and the proposed time baseline PG policies.**

of blocking external junctions, lowest number of halted cars and lowest waiting times. In comparison with the 4 phase rate aware policy the average waiting time per vehicle is reduced by a factor of ten and $CO_2$ emissions are decreased by more than 5.5 tons in just one hour. There is a small difference in terms of network capacity among the RL policies, as can be seen also in their very close performance in Figure 7 for cumulative departed vehicles in the whole network, which includes the highway traffic where the blocking queue rate has instead a big impact. The PG average baseline is also obtaining the highest return, however this occurs at the detriment of blocking other junctions with long queues more often, resulting in less completed trips and lower mean speed.

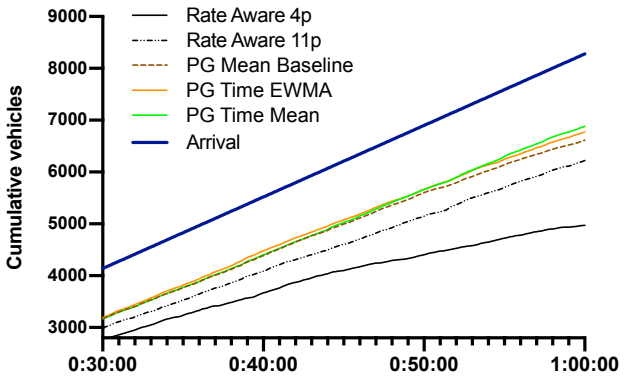


**Figure 7: Last 30 minutes of arrival and departure of vehicles in the whole network for the 5 policies in Table 3**

**Training efficiency.** We run 10 simulations for 500 episodes each to show the convergence properties of the proposed time baseline, for the arithmetic mean and exponential weighted moving average, in comparison with using the average return baseline. Results in Figure 8 shows the performance of the three PG methods.

## 6 RELATED WORK

In the past decades several timing models have been proposed with the goal of optimizing traffic signal control. These models usually assume that the arrival and departure rates of vehicles are known and build an analytical model to derive the time of the phase [21].

Several fuzzy logic controllers have also been proposed to respond to real-traffic demand [12]. Recently, novel computational approaches such as genetic algorithms [23], micro-auctions [6] and RL algorithms have shown promising improvements.

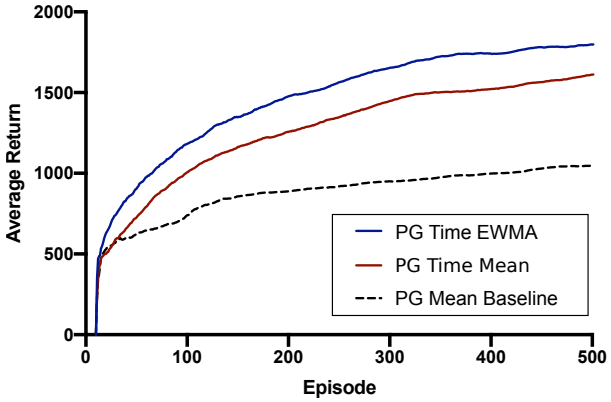


**Figure 8: Average return in the first 500 episodes, using the same real traffic data of the peak hour for each one.**

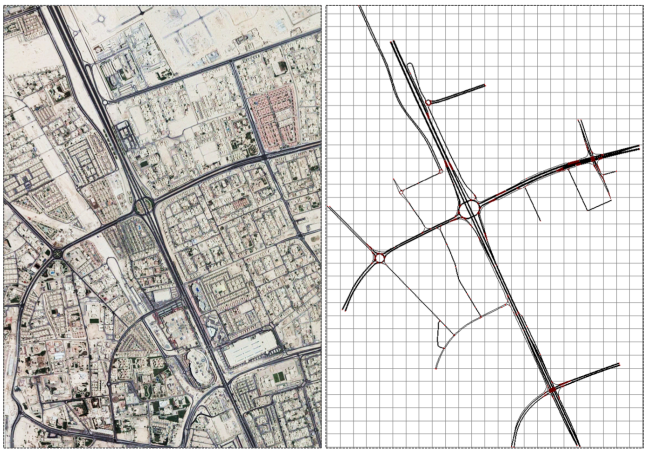

**Figure 9: The Al Gharrafa roundabout in Doha, viewed from satellite imagery from ©2019 DigitalGlobe, and the corresponding SUMO network. Each square side of the grid is 100m long.**

**RL approaches.** The optimization of traffic light control strategy has a long history in reinforcement learning, with one of the earliest work applying SARSA on the traffic light control problem dating more than two decades ago [24]. Reinforcement learning

approaches have modeled the MDP state using different representations, such as the discretized vehicles position [4, 27] or a range of traffic volume [2]. More often the traffic state is represented as the occupancy of discretized cells in a grid [7, 9] or the vehicles properties in each grid's cell [11]. Estimating the value of states with tabular approaches requires the exploration of a very high number of trajectories over the distribution of states and actions, and enough computer memory to store the value of each state. For this reason, classic RL approaches are computationally bounded by a limited state and action spaces. In [1] for example, tabular Q-learning is applied to a heavily congested environment using the lengths of the four approaching queues as state and the change of phase as action.

**Deep RL approaches.** More recent approaches use neural network as function approximators to estimate the traffic states value (Deep Q-learning [7, 8, 10]) or to learn directly the policy, as function from state to actions phases (Deep Policy Gradient [16]). Deep learning approaches allowed the use of more complex states representation, such as camera view from above the intersection or, more precisely, a screenshot of the simulator [16, 26].

Several studies have also explored the possibilities of simultaneously controlling multiple intersections using a distributed multi-agent system by communicating action advices [23, 27] or by using coordination graphs [9, 25].

**Roundabout modeling.** Studies that addressed the peculiarities of roundabouts had the main goal of modeling traffic delay and roundabout capacity. The High Capacity Manual 2010 [15] is a well known resource for guidelines, constants and formulas for capacity and delay estimation, which has been improved using a conflict matrix and queue theory by a more recent work [19] for 1-lane and 2-lanes roundabouts. Attempts at optimizing timings in signalized roundabouts have also been made, using pre-defined models [13] and the Cross-entropy method for Reinforcement Learning [14].

## 7 CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a policy gradient method based on episodic conditions and a time-dependent baseline to learn an optimal policy for traffic signal control in congested conditions. We applied the method on a realistic and complex scenario of an oversaturated roundabout feeding peak hour traffic volumes into the environment. Results using real data show that the learned policy significantly reduced the total waiting time and emissions, while being able to avoid traffic jams in connected junctions. We have also shown that in the setting of a positive reward and varying length episodes, the proposed time baseline for the Policy Gradient REINFORCE algorithm has better convergence properties then the traditional baseline based on average return. We plan to implement the policy in real setting and discussions with local traffic authorities are underway.

## REFERENCES


[1] Baher Abdulhai, Rob Pringle, and Grigoris J. Karakoulas. 2003. Reinforcement Learning for True Adaptive Traffic Signal Control. *Journal of Transportation Engineering* 129, 3 (2003), 278–285.

[2] PG Balaji, X German, and Dipti Srinivasan. 2010. Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems* 4, 3 (2010), 177–188.

[3] N. S. Bedrossian. 1992. Approximate feedback linearization: the cart-pole example. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*. 1987–1992 vol.3.

[4] Li Chun-Gui, Wang Meng, Sun Zi-Gaung, Lin Fei-Ying, and Zhang Zeng-Fang. 2009. Urban traffic signal learning control using fuzzy actor-critic methods. In *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, Vol. 1. IEEE, 368–372.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2018. Quantifying Generalization in Reinforcement Learning. *arXiv preprint arXiv:1812.02341* (2018).

[6] Michele Covell, Shumeet Baluja, and Rahul Sukthankar. 2015. Micro-auction-based traffic-light control: Responsive, local decision making. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, 558–565.

[7] Juntao Gao, Yulong Shen, Jia Liu, Minoru Ito, and Norio Shiratori. 2017. Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network. *arXiv preprint arXiv:1705.02755* (2017).

[8] Wade Genders and Saiedeh Razavi. 2016. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142* (2016).

[9] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs. In *Machine Learning and Knowledge Discovery in Databases*, Walter Daelemans, Bart Goethals, and Katharina Morik (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 656–671.

[10] Li Li, Yisheng Lv, and Fei-Yue Wang. 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* 3, 3 (2016), 247–254.

[11] X. Liang, X. Du, G. Wang, and Z. Han. 2019. A Deep Q Learning Network for Traffic Lights' Cycle Control in Vehicular Networks. *IEEE Transactions on Vehicular Technology* (2019), 1–1.

[12] Zhiyong Liu. 2007. A survey of intelligence methods in urban traffic signal control. *IJCSNS International Journal of Computer Science and Network Security* 7, 7 (2007), 105–112.

[13] Wanjing Ma, Yue Liu, Larry Head, and Xiaoguang Yang. 2013. Integrated optimization of lane markings and timings for signalized roundabouts. *Transportation research part C: emerging technologies* 36 (2013), 307–323.

[14] Mike Maher. 2008. The optimization of signal settings on a signalized roundabout using the cross-entropy method. *Computer-Aided Civil and Infrastructure Engineering* 23, 2 (2008), 76–85.

[15] Highway Capacity Manual. 2010. HCM2010. *Transportation Research Board, National Research Council, Washington, DC* (2010).

[16] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. 2017. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems* 11, 7 (2017), 417–423.

[17] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. 2017. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*. 2775–2785.

[18] H. Qian, K. Li, and J. Sun. 2008. The Development and Enlightenment of Signalized Roundabout. In *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Vol. 2. 538–542.

[19] Zhaowei Qu, Yuzhou Duan, Hongyu Hu, and Xianmin Song. 2014. Capacity and delay estimation for roundabouts using conflict theory. *The Scientific World Journal* 2014 (2014).

[20] M. Riedmiller, J. Peters, and S. Schaal. 2007. Evaluation of Policy Gradient Methods and Variants on the Cart-Pole Benchmark. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. 254–261.

[21] B. De Schutter and B. De Moor. 1998. Optimal Traffic Light Control for a Single Intersection. *European Journal of Control* 4, 3 (1998), 260 – 276.

[22] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[23] Kenneth Tze Kin Teo, Wei Yeang Kow, and YK Chin. 2010. Optimization of traffic flow within an urban traffic light intersection with genetic algorithm. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2010 Second International Conference on*. IEEE, 172–177.

[24] Thomas L. Thorpe. 1997. *Vehicle Traffic Light Control Using SARSA*. Technical Report. Online]. Available: citeseer.ist.psu.edu/thorpe97vehicle.html.

[25] Elise Van der Pol and Frans A Oliehoek. 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)* (2016).

[26] 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, New York, NY, USA, 2496–2505.

[27] Marco Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*. 1151–1158.

[28] 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. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 34.

# SUPPLEMENT FOR REPRODUCIBILITY

## Simulator parameters

The simulator provides a wide set of parameters which strongly affect the results. We change some of these parameters from their default values with the aim of ensuring a fair and realistic evaluation, which we provide below for the sake of reproducibility. Exact command line arguments are given in the Table below.

- Time to teleport is disabled. This option, enabled by default (set to 300 seconds), teleport to the next edge the vehicles waiting for too long in front of an intersection. We disable it as it gives unfair advantage to poor policies.
- Collision vehicles are removed. In multi-lane network with weaving junctions such as the evaluated roundabout and congested scenario, collisions occur frequently in SUMO. If no action is taken, vehicles may stuck and block the traffic. On the other side, teleporting the vehicle would consider the vehicle as departed, increasing the capacity measure and positively reinforcing collisions. We therefore set SUMO to remove the involved vehicle.
- Collision min gap set to 0, to avoid the event of a collision unless the vehicles physically overlap.
- Collision check junctions is enabled, to also check collision between vehicles on the same intersection.

| Prameter | Value |
|---|---|
| `--collision.action` | remove |
| `--collision-mingap-factor` | 0 |
| `--collision.check-junctions` | true |
| `--time-to-teleport` | -1 |
| `--step-length` | 0.5 |

**Table 4: SUMO parameters.**

The environment has two important parameters: the environment time step, that is the period of time before the agent can take a new action, and the actual SUMO time step, that is the actual step in the simulator. Because the SUMO time step represents the minimum reaction time, we set it to 0.5 seconds to reduce collisions and ensure smooth traffic flow, while environment step is 10 seconds long so that each phase can last a multiple of 10 seconds. When the new phase is different from the previous phase, this period of time includes a 3 seconds yellow light designed according to traffic laws and safety principles.

## Neural network and PG

We describe here hyper-parameters and details of the reinforcement learning framework.

- Neural Network: Implemented using PyTorch. Fully connected network with ReLu activation function, one hidden layer of 512 nodes, 68 input nodes (one for each detector) and 11 Softmax outputs (one for each action).
- Entropy: entropy is multiplied by an hyper-parameter weight, set to 0.001, before discounting the loss.
- Learning rate: Given the high scale of rewards, learning rate $\alpha$ is set to 1e-5.
- Actions selection: Action is selected stochastically in all the experiments including evaluation, using the output the neural network for the probability distribution over actions.
- Discount rate for computing the cumulative discounted reward is set to $\gamma = 0.99$.

## Episodic conditions

- To detect the blocking queue conditions, queues of halting cars are measured in specific lanes so that the condition is triggered when the queue reaches one of the four limit points, shown in Figures 10 to 13. Only the lane that causes interfering is monitored, that is, the one adjacent with the external edge.
- Maximum waiting time before triggering the episodic condition is 300 seconds, which is the same amount of time that triggers teleportation in default SUMO parameters.
- Green idling is checked monitoring the last 6 steps, as vehicles require some time to circulate and exit from the roundabout in traffic congestion. We use the exit detectors, near to the roundabout, to monitor the exiting cars.

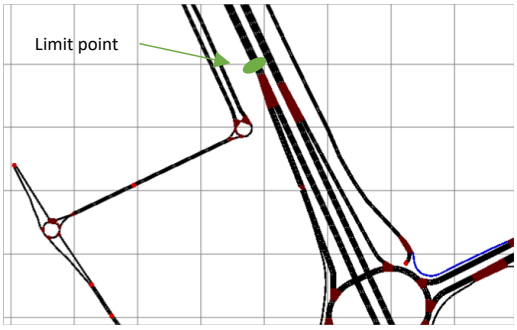


**Figure 10: Limit of halting queue in North entry arm.**

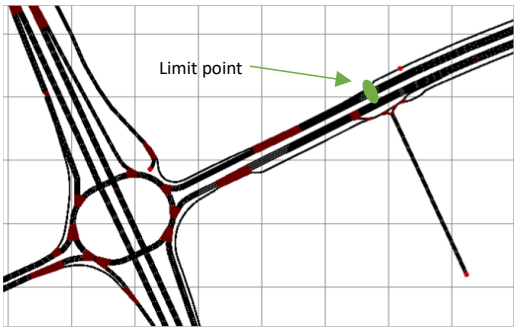


**Figure 11: Limit of halting queue in East entry arm.**

## Evaluation measures

To gather the evaluated measures we monitor all the edges approaching the roundabout and the circulatory lanes inside the

| Measure | SUMO TraCI method | Aggregation over edges | Aggregation over time steps |
|---|---|---|---|
| Complete trips | `simulation.getArrivedNumber()` | Not on edge | Sum |
| Mean Speed | `edge.getLastStepMeanSpeed()` | Mean | Mean |
| Mean Halted cars | `edge.getLastStepHaltingNumber()` | Mean | Mean |
| Waiting Time | `edge.getWaitingTime()` | Mean | Mean |
| $CO_2$ | `edge.getCO2Emission()` | Sum | Sum |

**Table 5: SUMO TraCI methods for evaluated measures and related aggregations.**

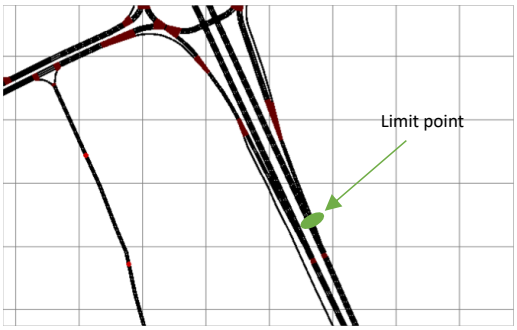


**Figure 12: Limit of halting queue in South entry arm.**

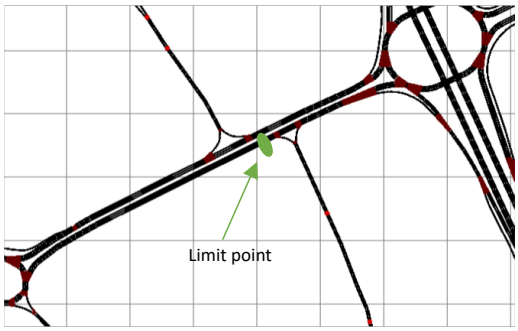


**Figure 13: Limit of halting queue in West entry arm.**

roundabout. Details on the aggregation of each measure over edges and time steps are detailed in Table 5.