# Traffic Signal Control benchmarks

sow5

December 2020

**Abstract**

Uprising for reinforcement learning in TSC. Explain quickly why. But no benchmarks papers so far. We introduce benchmark envs , reward, state, and results. We show that SOTA MARL algorithms fail at learning good policies.

# 1 Introduction

# 2 Background

## 2.1 The traffic problem

### 2.1.1 Metrics

### 2.1.2 Rule-based agents

**Random, Fickle, Min, and Max** Min: (Second-based) Change after min time elapses

Max: (Second-based) Change after max time elapses

Random: (Second-based) Choose a random action every second. Almost identical to Min policy to

Fickle: (Second-based) Choose a random time between min and max inclusive to change

Random and Fickle (pie-based): Make a random pie

**Offset Agent**

**Max pressure**

### 2.1.3 Limits

Rule-based agent can't adapt, what if demand suddenly change? What if there is no platoon? What about TSC at night? —- Reinforcement Learning.

Also we need coordination + Curse of dimensionality in action space + infrastructure problem —- multi agent RL

In the next section we will introduce what we need to overcome those limits...

### 2.2 Multi-agents Reinforcement Learning

## 3 Traffic networks

In some traffic configurations, where for example, the demand follows a simple pattern, rule-based agents might be enough while in other configurations, adaptable learning agents might be a necessity. As a result, to properly evaluate our agents, we design a predefined collection of traffic environments. The variable parameters are the demand and the layout.

**Demand** We propose four types of demand patterns:

- Platoon (P): a continuous flow of cars if appearing for a short period of time. In this situation, offset approaches should be optimal.
- Bernoulli (B): a every time step, a car is appearing randomly following a Bernoulli distribution.
- Gaussian (G): a every time step, the parameter of the aforementioned Bernoulli distribution is sampled from a Gaussian distribution.
- Poisson (S): cars are generated randomly, the waiting time between a car and its following car is distributed exponentially. The result is a number of independent Poisson processes (one for each edge that generates cars)

In real situations, demand might be changing over the time of the day. So we will distinguish between stationary and non stationary demands. To keep things simple, we define three types:

- Stationary (S): the demand stay the same for the whole simulation.
- Rushes (Ru): we simulate the morning and night rush. The whole simulation follows one of the aforementioned demand, but at two predefined times (far away enough to think of morning and night rush) of the simulation, there are outbursts of traffic.
- Random Rushes (RuR): same as NS, but the two outburst times are randomly distributed at the beginning of each simulation. In this situation, tracking the current time is not enough.

In some part of real networks, some links are subject to heavy traffic more than others. To acknowledge the asymmetry in the demand, we differentiate three (four?///) situations:

- Unique (U): the traffic comes from a single link, with a unique direction.
- Master-Slaves (MS): same as unique, but at rare occasions, cars come from other endpoints.

- Full (Fu): the demand is the same in all directions and all links.
- Mixed (M): demand depends on the direction (may be zero in some directions)

**Layout** In order to test coordination of multiples agents, we design networks with several intersections. We propose to create a fake (F) grid-shape network with $n \times m$ intersections. The grid can be asymmetrical such that an lucky initialisation of the traffic lights and a good offset cannot be enough to solve the coordination problem. We denote Sy for symmetric and ASy for asymmetric.

Finally, we design a layout based on a real network (R). The demand is predefined by a demand matrix. As the layout is fixed, we can only change the number of intersections $n$.

Here we list some examples of environments:

- F_Sy_1x3_S_MS_P: fake symmetric grid-shaped 1x3 network with stationary and master-slave platoon-like demand.
- R_3: real-based network with 3 intersections. We cannot configure the demand as it comes from a demand matrix.

Some pre-configured environment exists, number is increasing w.r.t. the difficulty; for example benchmark_0, benchmark_1 ...

# 4 Agents

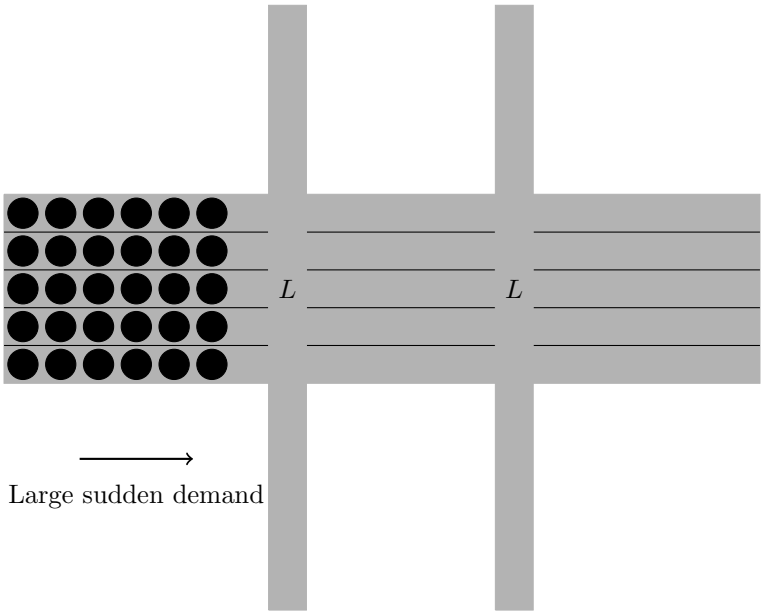
# 5 Experiments

# 6 Related work

# 7 Conclusion

# A Several ideas for more granular artificial benchmarks

## A.1 Zerg-rush




$L$ denotes a traffic light and a circle denotes a car.

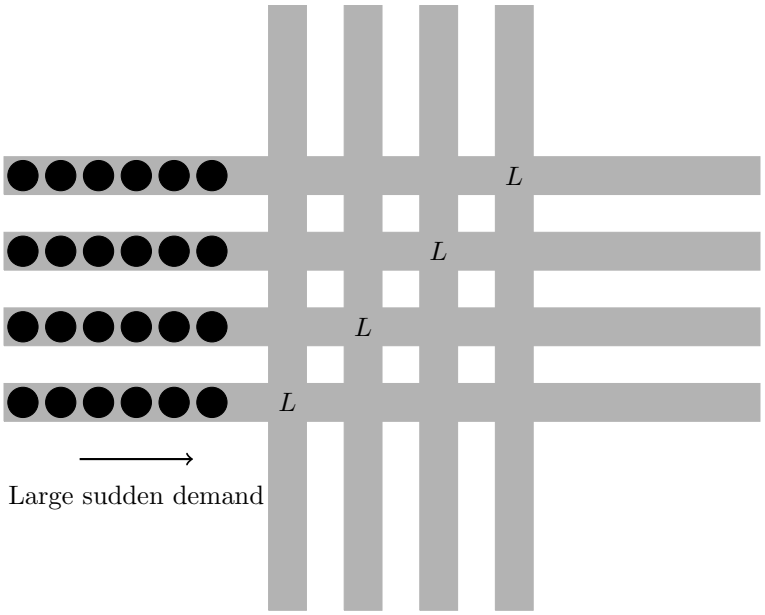
Multi-lane road with two intersections and sudden large demand impulse.

Place $n$ traffic lights, and test coordination (what state information is available to the lights? How many of their neighbours do the lights have access to?). Do the traffic lights coordinate to make a green wave?

Make variants with different timing of cars arriving at the side-roads.

Try periodic and aperiodic impulses.

## A.2 Zerg-rush-with-offset




$L$ denotes a traffic light and a circle denotes a vehicle.

Each light should have access to the state of each other light, but they should all have a limited cone of vision. Also, each light should see all four roads.

Tests cooperation: Do the later traffic lights receive information regarding the rush from the earlier traffic lights?

Time the demand on the verticals so that the optimal behaviour is to learn an offset.

## A.3 Single intersection

### A.3.1 Uniform or Poisson demand

The optimal behaviour (I think: Ilia) is that the proportion of time the light spends green NS to the time of light spends green EW should be equal to the proportion of demand NS to the demand EW.

### A.3.2 One-way demand

If there is demand coming one way, the traffic light should learn the max policy for that direction and the min policy for the other direction.

## A.4 Circle

Similar to one the Flow benchmarks. Can we coordinate cars on a circle to spread apart using several traffic lights instead of a control car?