Reproducing results from the Urban Traffic Control paper

Jakub Filipek (balbok)

December 16, 2020

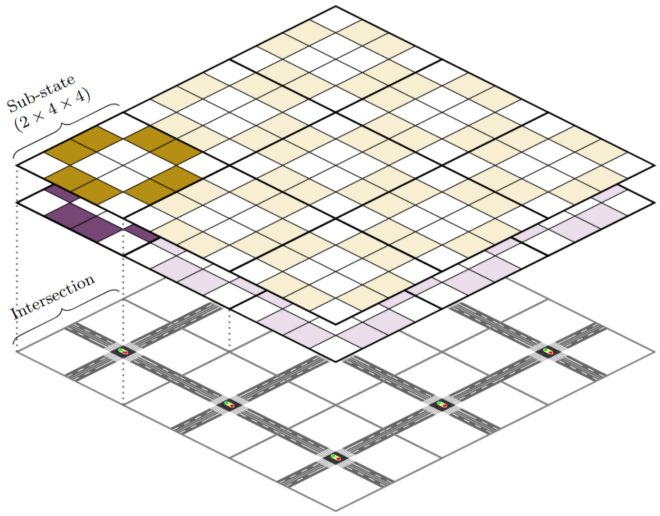
1 Introduction

The Urban Traffic Control paper [1] is interesting because it combines Reinforcement Learning with more-complex cases of Urban Traffic Control (UTC). UTC is hard because state spaces are vast and relations between traffic lights are not well understood. Furthermore, there has not yet been a good example of RL approach solving systems of intersections with complexity similar to large city centers.

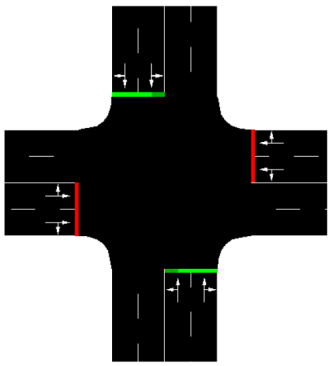
While [1] doesn't necessarily solve the second issue, the proposed method scales well with number of intersections, which may generalize to much more complex systems than 3-by-3 grid explored in original paper, and this reproduction attempt.

2 Environment

2.1 State Space



(a) Network to state space conversion.



(b) Intersection visualization in SUMO.

Figure 1: State space representation. On the left, each traffic light is converted into a $2x4x4$ tensor, were detector from each incoming lane contributes two features. The right shows a SUMO representation of a single intersection for reference.

The state space is $R^{2\times12\times12}$. It consists of 9 (3×3) $R^{2\times4\times4}$ segments, each representing a traffic light. For each traffic light there are 150m detectors (not visualized) on each of the incoming lanes (500m each). For each lane two features are captured (mean speed of vehicles on the detector, number of halted vehicles). Hence there is sixteen features for each intersection. They are aranged into a $2 \times 4 \times 4$ space, with first dimension for the two features, and 4×4 grid capturing the orientation of each incoming lane (North, West etc.).

Fig. 1 shows visualization of the state space, as presented in [1].

2.1.1 Additional state features

One perhaps missing part in state representation is the state of traffic lights themselves. In original work, the agent has no information whether which phase (See Fig. 2) the traffic lights are currently on.

To address this concern I added additional feature for traffic light state. As every incoming lane has two outlet lanes (left/straight or straight/right), this feature is defined as a sum:

$$f(i) = \sum_{o \in OUT(i)} \begin{cases} 0.75 & \text{if } o \text{ is green} \\ 0.25 & \text{if } o \text{ is yellow} \\ 0.0 & \text{if } o \text{ is red} \end{cases}$$

where i is incoming lane and o is a state of traffic light between i and some outlet lane (given by relation OUT).

Hence in this case the resulting state space is $R^{3\times12\times12}$.

2.2 Action Space

The Action Space is $\{0, 1\}^9$. For each traffic light the phase can be changed (1) or remain the same (0).

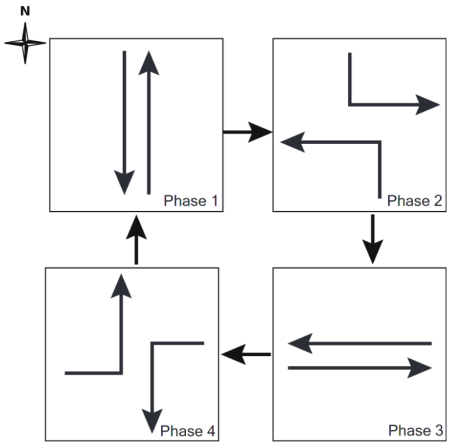


Figure 2: 4 phases of a traffic light. The light oscillate between straight and left turn, and NS/WE directions. Right turn is always on, as it does not collide with any other movements. There are also yellow light phases corresponding to each of the arrows on the figure.

There are two caveats however. First is that traffic light cannot be changed if it has any yellow lights. This is done mostly due to the simulation, where cars can perform instant breaking with no reaction time, thus making it unrealistic. Setting yellow light phases time to 3 seconds makes this reproduction attempt avoid this problem and be consistent with [1].

Secondly, lights can change on its own, which is an artifact of SUMO. This is most important in case of yellow lights, as automatic iteration is the only way phase can change in that case. In all other cases agent can invoke change of light (with 1), but if enough time has passed SUMO will change a light due to cycling of a light. To avoid this problem I set phase duration for non-yellow light phases to 600 seconds. Without agent (action is always 0^9) network becomes clogged. Such behavior is crucial for agent to not depend on pre-set cycles, but it is not at all described in [1] and might lead to differences in the results.

2.3 Reward function

The reward function involves both global part, which purpose is to evaluate throughput of the network, and local part, which balances intersections.

The global reward function is:

$$R_g = |V_{out}| - |V_{in}| \tag{1}$$
$$\tag{2}$$

where V_{out} and V_{in} are numbers of outgoing and incoming vehicles, respectively. The vehicles which teleported out of the network are not included, making calculation consistent with original work.

The local reward function is:

$$R_l(i) = -|\max q_i^{\mathrm{WE}} - \max q_i^{\mathrm{NS}}| \tag{3}$$
$$\tag{4}$$

for each intersection i. q_i^{WE} denotes lengths of queues for east- and westbound traffic. Similar definition is used for q_i^{NS}.

Overall, combining Eqs. 2 and 4 we get:

$$R = \beta R_g + (1-\beta)\frac{1}{N_{TLS}}\sum_{i}^{N_{TLS}} R_l(i) \tag{5}$$

where β changes linearly during training from 0 to 1. This ensures that agent first solves local problems, then moving to optimization of global network.

3 Agent

Both the Actor/Critic model and PPO+GAE algorithm has been well described by the original paper in Sections IIID and IIIE, respectively.

3.1 ResNet Based CNN

While 3 describes architecture well, both it and the original text omit some hyper-parameters. Size of kernels and channels for Convolutions in non-ResBlocks has been omitted. In this case for the first such layer I used a 3×3 convolution with 32 channels. For remaining convolutions I used a 4×4 kernel, motivated by size of single traffic light in state representation, along with 32 channels.

Another unspecified hyperparameter was in size of fully-connected layers, which I set to be $(256, 32), (32, 1(\text{Critic}) \text{ or } 2(\text{Actor}))$.

Lastly, for sake of verbosity, the output shapes corresponding to variables on Fig. 3 are as follows:

$$a \in R^{9\times 2} \qquad v_{local} \in R^9 \qquad v_{global} \in R$$

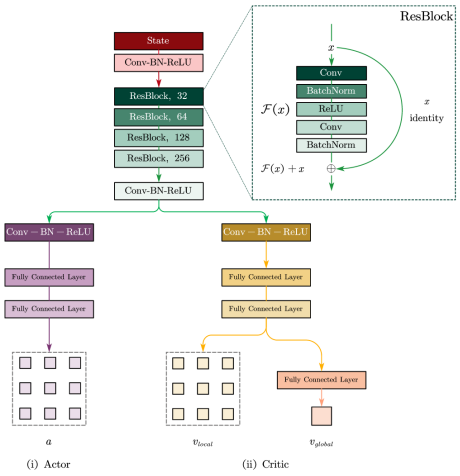


Figure 3: Visualization of the CNN architecture as described in [1].

3.2 PPO + GAE

Proximal Policy Optimization (PPO) [2] has proven to be a widely successful algorithm in variety of problems. As the original (UTC) paper claims its monotonic improvement through probability ratios is preferred in case of complex systems such as set of traffic lights.

The description is clear and exact in the original work (Section IIIE), I will not repeat the description of common PPO parameters. However for reference the parameters are shown in Table 1. The α parameters changes linearly from 1 to 0 during training.

Hyperparameter	**Value**
Horizon (T)	64
lr	$\alpha 10^{-4}$
# Episodes	50
# Epochs	3
Mini-batch Size	1024
γ	0.99
λ	0.95
# Actors	16
Clipping parameter ϵ	0.1
Entropy coefficient	0.01

Table 1: Hyperparameters of PPO algorithm. It's almost a 1-to-1 copy of the original work, except for lack of α in clipping parameter.

The only modification from original work comes in lack of α in clipping parameter. However, given that PPO already considers ratios rather than absolute changes it should not be influential.

Additionally, each environment takes 3600 steps before autoreseting itself with new vehicle routes, which are generated using built-in trip generator following parameters described in original work. This corresponds to one episode in original work (though it is unclear whether paths are changed between episodes). In my work one episode contains additional 128 steps for rollout.

4 Coding Environment

The work was done utilizing SUMO [3] for the traffic simulation. OpenAI Gym [4] environment was used on top of that with custom code.

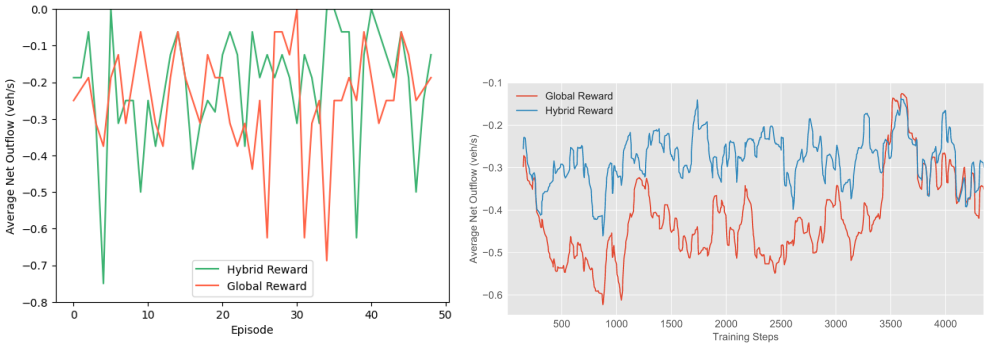
For PPO+GAE algorithm RLLib [5] was used with no modifications. An attempt at implementing own version of the algorithm was made, and while successful in terms of early training it was too slow and infeasible to finish experiment.

All of the code is available here.

5 Results

Overall, three models have been trained. One for hybrid reward, one for global reward, and one for hybrid reward with state of traffic lights as a feature.

5.1 Global vs. Hybrid rewards



(a) This project. (b) Original work.

Figure 4: Median training reward per episode of models from this project and original work. While not the same, they both show values on similar scale, with both hybrid and global reward versions having between -0.1 and -0.3 final reward. The first episode reward is omitted for scaling purposes.

As [3] claimed hybrid reward lead to more stable behavior during training, which can be see on Fig. 4b. However, I was not able to reproduce such a behavior. Overall, both of my models were more volatile, which can be due to difference in clipping parameter schedule. Another possible reason is changing routes of vehicles every episode which may lead to less stable behavior inter-episodes.

The reward ranges are similar between the two trainings showing that while reproducing relative performance relations was not achievable, the overall structure of the model is clear enough to train similarly performing agent.

5.2 Traffic Lights as a feature

As discussed in Section 2.1.1 addition of traffic lights seems to be a reasonable decision considering that it is a part of state space directly affected by actions. However, as shown on Fig. 5 there is no significant difference in performance between agents using and not using this feature.

The feature counting number of halted vehicles might be a good proxy for which lights are currently turned on and off, causing the two to be highly correlated. This could lead to no differences in adding such a feature.

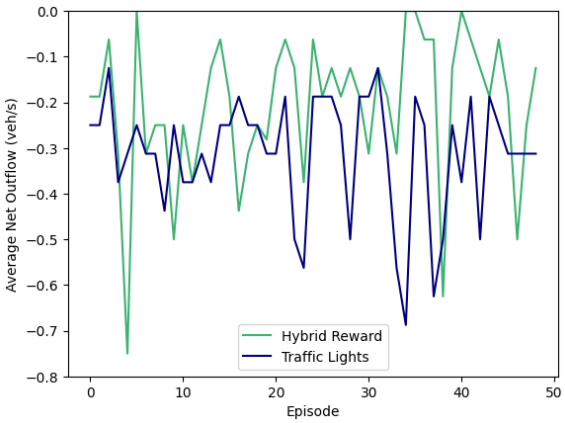


Figure 5: Median training reward per episode of models with and without traffic lights. The first episode reward is omitted for scaling purposes.

The one perhaps desirable feature of adding traffic lights state as a feature is seemingly more stable behavior across episodes. As routes change every episode for every agent, the variance in results is to be expected. However, with different route patterns the correlation between halted vehicles and traffic light state might differ, in which case having both of the features stabilizes training. This would reduced inter-episode variance as seen on a graph. However, it can also be due to randomness, and would have to be repeated over many runs to make substantial claims.

6 Conclusion and Future Work

Overall, I was able to reproduce majority of the results from the original work [3]. The main point I failed to reproduce was the relation between hybrid and global reward, which can be due to differences in training processes.

As mentioned in Section 1 the main advantage of this approach is the efficient scaling of the state space with the number of traffic light. I would like to further test it on variety of real-life environments. In particular the 500m distance between each pair of intersections seems rather large for most city centers, and training a model on a denser, busier and overall more challenging environment might give it more room for improvement.

References

[1] Y. Lin, X. Dai, L. Li, and F.-Y. Wang, "An efficient deep reinforcement learning model for urban traffic control," 2018.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[3] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[5] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018.