WEBLIGHT: DRL based Intersection Control in Developing Countries with Reliable Cameras

Anonymous Author(s)
Affiliation
Address
email

Abstract

Effective traffic intersection control is crucial for urban sustainability. State of the art research seeking Artificial Intelligence (AI), for example Deep Reinforcement Learning (DRL) based traffic control requires environment states through various Computer Vision methods, where the collective state of multiple cameras across an intersection constitute the single state for AI. This brings in serious robustness or fault-tolerance concerns on the deployed system. Camera systems are highly susceptible to faults due to multiple possible points of failure. A single fault collapses the AI state and hence the capacity of AI controller to manage the traffic is gone. Also, infrastructure deployment and maintenance is a slow bureaucratic process in these countries, which makes camera faults a regular event. In the given paper, we build a web based, independent and alternative, traffic state processing method which can replace the camera dependency completely, or support as a backup mechanism until the camera system is back online, making the AI intersection control robust to camera failures.

1 Introduction

While gathering the real data for an intersection for analysis, we observed multiple issues hampering the sound operation of the deployed cameras. Some of the issues were related to

- One camera power adapter failure
- Power failure for one approach
- Communication line failure from one approach
- Multiple times Local Processing Unit hang or power-off

In issues related to the camera device, a crane was required for repair. For issues at ground level, efforts have to be made to trace down the point of failure, and then replace the faulty component. All of these require days to weeks to get done, due to many dependencies involved in the maintenance process. Due to such problems, we could get just 40 days of complete data in 4-month duration.

We started to wonder how can AI based systems be deployed effectively with so many failure points present in even a simple camera based traffic control system. A fault in a single component can collapse the whole AI state, and the capacity to control the traffic is gone. To overcome this problem, we analyzed for an alternative mechanism to compensate for camera failures until the repair is done. This requires getting traffic dependent travel time(s) across the intersection from Webservice APIs provided by Google or Microsoft. For our analysis, we used the Distance Matrix (Advanced)[1] API of the former , which returns the distance with real-time travel times for the set of points given.

There maybe a side concern that a good and consistent network connectivity is needed for the proper functioning of web-based methods. For such scenarios, we will only miss receiving latest information for the time instant where connectivity goes low. We can utilize the information received in the

[1] https://developers.google.com/maps/documentation/distance-matrix/overview

Submitted to Computational Sustainability Workshop at NeurIPS 2023. Do not distribute.

Type	Duration	Desc	Freq	Cnt	Traffic Density			Google Time		
					Min	Mean	Max	Min	Mean	Max
Train	8 hours	Daytime	5 sec	5760	0.011	0.360	0.935	35	72.04	188
Test	4 hours	Till-noon	30 sec	480	0.013	0.323	0.931	47	74.79	117

Table 1: Dataset Description with absolute 3M (Min,Mean,Max) Traffic Density, Google Time values

previous communication as an approximate. Also, the connectivity will resume automatically without our intervention, which is not the case with installed cameras as something has to be done to fix their failures. Also, as per our analysis and observations from the traffic API, the real-time travel times does not fluctuate highly over a few seconds. Related work is discussed in Appendix A.

2 Data Description

To analyze the suitability of Google Time data in place of Traffic Density, we gathered two sets of data, an 8 hour high frequency (5 sec) data for training and analysis, and a 4 hour low frequency (30 sec) data for evaluation. As the purpose of evaluation data is to validate the generalization of the learning over a longer period, high frequency test data is not cost-effective. Various other details on these data are given in Table 1. This process required data collection from two modules in a time synchronized manner -

i. Traffic Density: This data was gathered using Bouwmans (2014) from the cameras placed at a busy intersection in New Delhi.

ii. Google Time: We requested real-time data from Google for the various approaches at the intersection. Each time value correspond to the primary flow of traffic for each phase for 400mt distant points across the intersection.

We pair the Traffic Density and Google Time values based on the frequency of the latter. We ignore the intermediate Time Density values in that duration.

3 Evaluation Baselines, Benchmarks, Metrics

To validate the performance of Google Time, we performed experiments over various scenarios and traffic datasets, and compared the performance with suitable baselines.

3.1 Baselines

We compare our methods with state-of-the-art RL models and recognized baselines:

i. *Presslight* (Wei et al. (2019)) for decentralized multi-intersection processing with 20x20 DNN architecture and MaxPressure as the reward.

ii. *FrugalLight* for low-resource decentralized intersection processing, with 15x15 DNN architecture with StopDensity as the reward.

iii. *EcoLight* (Chauhan et al. (2020)) for efficiently deployable decentralized intersection control, with 10x10 DNN architecture and StopDensity reward.

iv. *Fixed Timing:* for signal switching to the next approach after fixed time intervals.

Our $RLDecision$ module contains 15x15 DNN architecture and Stop Density reward.

3.2 Benchmarks

We use 4 real road datasets (1-4 hour each) -

i. Dataset 1 is a legacy traffic data from same intersection, processed using YOLO Redmon et al. (2016) and used in our prior works.

ii. Dataset 2 is generated from the procured training data, using Traffic Density based processing.

iii. Dataset 3 is a new 4 hour dataset generated from the recent procured training data.

iv. Dataset 16x1 is a 16-intersection 4-approach dataset taken from the baseline Presslight Wei et al. (2019). As our real data based Google Time conversion supports 3 approaches, the experiments with this dataset consider the traffic on three incoming approaches.

3.3 Metrics

The average case performance metrics are i. *nOut*: the number of vehicles cleared, ii. *Travel Time*: time spent by cleared vehicles, and iii. *Total Time*: time spent by all vehicles, over the single or the network of intersections as per the dataset type.

3.4 Simulator

We use the CityFlow traffic simulator Zhang et al. (2019) for these experiments, for the purpose of comparison with suitable state-of-the-arts. It requires road network structure, phase information and traffic dataset. We can set the desired phase using API calls. For every phase change, a 5-second yellow and all-red time is given to clear the intersection. CityFlow provides environment information via various data-structures, which are processed to compute various metrics to compare across the control algorithms.

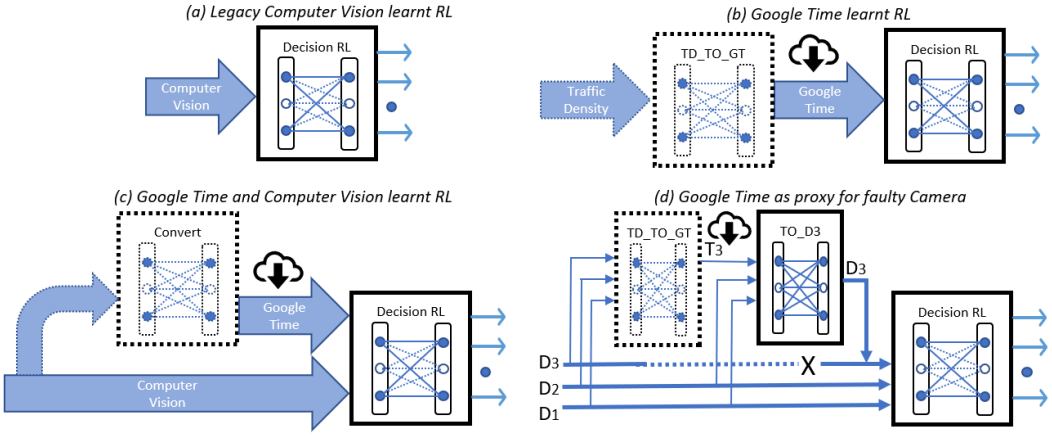

Figure 1: Various RL models (Dotted DNNs used only for training i.e. only solid DNNs are deployed)

4 GoogleTime based RL model performance

We generate TD_TO_GT conversion mapping, learnt by fitting Traffic Density to Google Time over a DNN, as per the process described in Section C. We explore a range of multiple DNN based models (in Figure 1) to seek a fault tolerant solution. We performed various experiments, for the 3 real datasets and the results are shown in Table 2, GT denotes Google Time and TD denotes Traffic Density. We perform RL training for 250 epochs and then average their metrics for the next 50 unseen epochs. We employ a DNN with loss function as Mean Square Error, with details shown in Table 4.

The Figure 1(a) shows the current way of exploiting RL models for traffic management, where camera data is processed via Computer Vision methods to get traffic measurement (like density). Figure 1(b) shows a way to exploit Google Time at training through simulation. During training, we convert the simulator provided Traffic Density to Google Time using the conversion learnt by TD_TO_GT module over real data. Once the model is trained, we can remove the TD_TO_GT conversion module and directly feed Google Time data taken from web to the $DecisionRL$ for real-time usage. This model is highly suitable for developing regions where a large number of intersections can be AI controlled without cameras installation and building other backbone infrastructure. Figure 1(c) gives a method to exploit both traffic density data and Google data to allow RL learn a more robust policy than existing methods with unary data. Here, the actual Traffic Density information is also given to $DecisionRL$ alongside Google Time.

Finally, Figure 1(d) gives a method to exploit Google Time data to compensate for the Camera with a fault, thus allowing RL to function as it would with the real camera data. The Google Time,

Fig	Model	DNN			1			2		
		State	Arch	Param	nOut	Travel	Total	nOut	Travel	Total
1(a)	Presslight	80	20x20	2082	1250.2	244.31	249.91	1763.2	158.79	162.16
	FrugalLight	3	15x15	332	1214.3	238.85	257.39	1808.5	177.02	177.52
	EcoLight	2	10x10	162	1246.7	253.83	254.16	1860.7	182.60	181.04
1(b)	GT	3	15x15	332	1146.8	113.09	264.18	1664.2	126.71	192.17
	GT(S)				1229.9	185.63	247.53	1795.3	149.13	173.25
1(c)	GT+TD	6	15x15	377	1291.2	240.65	241.75	1774.6	161.98	163.76
	GT+TD(S)				1295.4	249.47	241.72	1774.6	161.98	163.76
1(d)	GT+TD(B)	3	15x15	332	1220.3	195.92	251.77	1784.8	145.35	163.50
	GT+TD(BS)				1242.1	230.36	250.33	1784.8	145.35	163.50

Table 2: Performance of WEBLIGHT's RL models. Models corresponding to Fig 1(a) are the baselines

corresponding to the traffic movement(s) underlying the failed camera, is exploited to predict the traffic density over the failed camera's approach. To prepare the system with such capacity, we train each of TO_D1, TO_D2 to TO_Dn modules, for each approach, to handle the failure of any camera of that approach. During training over simulation, we use the TD_TO_GT conversion module alongside to facilitate the generation of Google Time corresponding to the faulty camera being considered. Finally, all of TO_D1 to TO_Dn modules and $DecisionRL$ module are deployed to final system, and respective TO_D module is selected automatically as camera fault is detected.

We observed that some intermittent models exhibit dull learning i.e. they either rarely switch to next phase, or seldom remain at a phase. This makes the model ineffective, and hence it can be discarded. The results after discarding these models and Selecting the remaining ones, are shown as GT(S). We believe that such behaviour occurs due to the outliers in the google data which get fused into the conversion models, and influence bad convergence for some intermittent models. The good thing is that it is easy to detect them by simply observing the signal change count. We also see that such dull learning is rare in 1(c) and 1(d) due to the influence of Traffic Density, where there is minimal change in metrics after the selection.

As seen in Table 2, the performance of WEBLIGHT models with Google data and RL methods is at par with the state-of-the-art RL based methods. It is a promising result showing the effectiveness of our methods as well, alongside the robustness benefits. These outcomes are highly encouraging for developing regions where large number of intersections can be AI controlled without the installation of cameras and building other backbone infrastructure.

In the models given in Figure 1, the dotted DNN represents conversion module which is used during training as a proxy to real time Google Time data. On deployment, it will be replaced with actual (realtime) google data. Also, as the $DecisionRL$ is trained on good correlated Google Time and corresponding Traffic Density data, its expected to perform better with the real data either directly or with limited fine-tuning.

5 Non-RL time-based methods performance

Similar to the utility in RL based models, Google Time can also be utilized to improve Time based policies. We explore this in three flavours - Phase, Cycle and Random. The *Phase* queries for Google Time for the new phase at the start of each phase. The *Cycle* queries the time of all phases at the start of each cycle. The *Random* queries the time on random basis over multiple cycles. For any flavour, we simply allow the traffic signal to hold on to the phase for the duration as given by Google Maps in the previous API call, and then switch to the next phase.

We performed experiments on the 4 real datasets discussed in Section 3 and employed the same TD_TO_GT modules as used for RL experiments, to facilitate Google Time values in simulation environment. The results are shown in Table 3. We can see that all Google Time variants perform better than a Fixed Time policy. For 16x1 dataset, we see that Travel Time is higher for our methods (shown in *italics*). Due to the linear arrangement of multiple (16) intersections and increased throughput, our methods allow some additional vehicles (with high travel time) to exit the network, thus pushing the overall average higher. The *Cycle* variant may be biased to a particular phase as it takes the Google Time values at the start of a particular phase. The *Random* variant performs fewer queries relatively, and is our recommendation considering the performance.

Method	1			2			3			16x1		
	nOut	Trvl	Totl	nOut	Trvl	Totl	nOut	Trvl	Totl	nOut	Trvl	Totl
FixedTime	1191	274	265	1586	257	249	5990	231	230	1672	*320*	1002
Phase	1361	230	224	1772	211	206	6408	188	187	1734	*326*	975
Cycle	1358	233	226	1726	235	228	6673	230	228	1834	*369*	922
Random	1356	233	226	1761	208	204	6352	193	192	1735	*360*	995

Table 3: Performance of WEBLIGHT's Time based (nonRL) models on 4 real datasets

6 Conclusion and Future-work

In the presented research work, we analyzed the Google Maps Time for its capacity to be utilized as a mechanism to control and manage the traffic signal, in order to sustain a fault-tolerant system. We can use approach-specific conversion models and intersection-level RL models to effectively bypass camera based failures. We also presented a Random query based mechanism to control traffic lights just by using the Google Time and without any camera based infrastructure.

Appendix

A Background and Related Work

The traffic problem has been widely acknowledged, and traffic control authorities are putting efforts to bring new and effective ways to manage the traffic flows. To reduce the waiting times at traffic signals, authorities convert the roads to one-way traffic int (2018b) and introduce new roundabouts int (2014) at traffic heavy intersections. These have been the traditional ways to manage the flow of traffic.

With the fast progression and blending of technology with public life, new technological opportunities have been created. The authorities are open to experiment on such avenues. Shifting from wired connections between traffic controller and the control lights, to wireless control is already in progress int (2018a). CCTVs can be used to find queue length (with initial infrastructure factor), underground induction loop sensors can count the number of vehicles (which will be effected by unwanted digging), and Google Maps can be utilized (which has its own set of concerns).

In a ground experiment by the city control authority at Hyderabad (India) int (2019b), Google Maps was utilized to increase the wait time for an approach if the traffic coming towards the intersection on that approach is low. Based on the trails, the method is expected to reduce waiting times by 30% and queue lengths by 50%. The authorities at Bengaluru (India) int (2019a) performed similar trial, and reported a 10-20 seconds improvement in wait times at a key intersection.

One research group Kumarage et al. (2018) analyzed Google Maps travel time information over long distances in an urban area in SriLanka for the purpose of a pilot project for having a priority lane for buses. They observed a small increase in the travel time, and a small decrease in the speed (then available via Google Maps API), during the trials. They also analyzed the travel time and speed information from Google to predict the traffic flow. The flow was observed through an InfraRed Traffic Logger system. They used the spacial features of the road network in the process, which might make the system less transferable.

In another work Arunachalam Muthupalaniappan (2019), the authors extracted the traffic information level from the colour codes of Google Maps image at a central server and then scaled the values to represent traffic data. Then they use a custom algorithm for controlling the traffic system.

For our analysis, we utilize the Google Maps differently, getting precise time data using the API over small distances across the intersection.

B Issues with Correlation

On analysis of the correlation between Traffic Density and Google Time (using DNN as in Table 4(a)), we observed a good correlation and encouragement for Google Time to be used effectively for the traffic management purpose undertaken by Traffic Density till now. Almost 80% data samples showed less than 15% prediction error. There was a small fraction of data with unreasonable deviation.

	Purpose	Type	Loss Function	Optimizer	Architecture
a	Correlate	Regression	Mean Squared Error	RMSprop	15x15
b	Classify	Classification	Sparse Categorical Crossentropy		FullyConnected

Table 4: Various DNNs used in the fault-tolerance analysis.

To find the cause of the small set of outliers, we analyzed the data to see if we can find some of the outliers without using AI methods. So, we utilized a method to make many bins quantizing the traffic density. The samples falling into each bin have Traffic Density in a small range, but the Google Time can be in any range. A small processing in the bins lead us to one form of outliers as shown in Table 5. Here, and for all subsequent experiments, we use the 400mt Google Time due to its alignment with Traffic Density.

For every block of samples, 3 samples are presented, spanning over small period of time. The middle sample is from the clock time between the other two samples, and carries Traffic Density in between the two samples as well, yet the Google Time (ground truth) values are far beyond the values from other two samples.

Data	Clock	Traffic Density			Google Time			Our Prediction			Percent Error		
	11:33:00	0.37	0.27	0.21	79	78	73	80.73	83.40	66.38	2.19	6.92	9.07
Test	11:43:00	0.35	0.25	0.04	**59**	81	**80**	77.98	80.06	63.58	**32.17**	1.16	**20.53**
	11:53:30	0.16	0.22	0.04	87	82	72	75.58	76.32	63.55	13.13	6.92	11.74
	13:45:00	0.18	0.14	0.20	80	80	69	76.69	75.93	65.33	4.14	5.09	5.33
Test	13:57:00	0.20	0.16	0.21	71	**111**	**101**	77.25	76.97	65.62	8.81	**30.66**	**35.03**
	14:25:30	0.25	0.17	0.22	82	82	68	77.97	77.97	65.73	4.92	4.91	3.34
	13:29:50	0.05	0.08	0.51	79	86	70	78.32	76.02	69.59	0.86	11.61	0.58
Train	13:58:00	0.04	0.07	0.49	72	**161**	69	77.73	75.11	69.18	7.96	**53.35**	0.26
	14:45:15	0.02	0.07	0.47	84	80	69	77.24	74.50	68.90	8.05	6.87	0.14
	16:03:25	0.10	0.04	0.46	81	70	70	77.38	74.15	68.41	4.46	5.93	2.27
Train	16:42:10	0.21	0.04	0.49	**67**	**85**	**115**	78.82	76.06	68.66	**17.64**	**10.52**	**40.30**
	16:49:30	0.24	0.04	0.55	84	77	77	79.91	77.35	69.53	4.87	0.46	9.70

Table 5: Out of sequence Google API response with respect to Traffic Density for the 3-approach intersection

Our learnt TD_TO_GT model predicts the Google Time with the same soundness for all Traffic Density values. These predicted Google Time values, which are biased largely towards the majority / correlated samples, are in a narrow range for all the three samples This also raises a concern, if this middle sample is valid, we do not see a way with respect to Traffic Density to learn this effectively. And if this is really an outlier, this must be impacting/disturbing the DNN learning, and removing it should be fruitful for the learning process.

C Filtering of Outlier data

We utilize a divide-and-rule based filtering process to refine the data-pairs with high level of correlation. In the process, the outliers are removed in a phased manner to improve the chances of good data-pairs being selected for further processing. An algorithm describing this process is described as below -

Algorithm 1 Divide and Conquer based Filter Process

Require: $TrainSet, TestSet$
$errStart \leftarrow 35$
$errEnd \leftarrow 15$
$errStep \leftarrow 5$
$model \leftarrow$ FitRegressor$TrainSet$
for $errLmt \leftarrow errStart$ to $errEnd$ by $errStep$ **do**
  $TrainSet, Outlier \leftarrow$ FilterData$model, TrainSet, errLmt$
  $model \leftarrow$ FitRegressor$TrainSet$
  $3Mstats \leftarrow$ EvalModel$model, TestSet$
  $classifier \leftarrow$ FitClassifier$TrainSet, Outlier$
  $TestOk, TestOutlier \leftarrow$ ClassifyData$classifier, TestSet$
  $3MstatsOk \leftarrow$ EvalModel$model, TestOk$
  $MaxSeq \leftarrow$ GetMaxSequence$TestOutlier$
  print $3Mstats, 3MstatsOk, MaxSeq$
end for

For $FitRegressor$ and filtering outlier data from the training set using $FilterData$, we use DNN with parameters as shown in Table 4(a). For $FitClassifier$ and classifying (unseen) test data using $ClassifyData$, we use DNN with a loss function as Sparse Categorical Crossentropy, with other features as shown in Table 4(b). The classification gives a confidence on the percentage of Google Time values to be effectively predicted using Traffic Density during training over simulation. $EvalModel$ is used for evaluating the trained models over the test data, and provide the 3M (Min, Mean, Max) metrics. The values for $errStart$, $errEnd$ and $errStep$ are selected by intuition for experimental purpose.

D Impact of Data Filtering and Classification

The results of the filter process are shown in Table 6. We can see that the 3M metric error for the raw test dataset keeps improving (reducing) as the training data is refined. We further opted to employ a classification mechanism to filter the test data as well, for which we see better prediction capability. Hence, this classification mechanism can be used as a confidence predictor to further show how reliable the prediction on unseen data can be. It also opens a window for new algorithms to be researched for the set of Filtered Outlier Test Data.

The process enables us to find a generic and improved correlation between Traffic Density and Google Time, which is depicted in the RL based experiments as well. These refined models are deployed as TD_TO_GT modules in Figure 1, which facilitate effective simulation based RL training for traffic policy convergence. Regarding the outlier correlation, we think that it maybe due to **i.** some historic learning of the Google Maps service (which influences it to consider different traffic volume than it is on the road), or **ii.** an outcome of the network dependent crowd-sourced data (which is tough to always be realtime and accurate).

Max	Train	Raw Test Data			Filtered Good Test Data				Outlier Test Data	
Error	Cnt	Min	Mean	Max	Cnt	Min	Mean	Max	Cnt	MaxSeq
35	89.11	0.015	10.11	42.61	99.17	0.015	10.01	42.20	0.83	2
30	81.84	0.009	9.61	41.24	95.21	0.013	9.18	36.78	4.79	2
25	71.65	0.011	9.13	41.27	87.71	0.011	8.21	28.86	12.29	6
20	59.05	0.009	8.43	40.11	74.79	0.010	6.81	23.21	25.21	12
15	43.13	0.007	7.84	38.74	57.92	0.010	5.39	21.44	42.08	17

Table 6: Percent Error after sequential filtering process

References

2014. 5 roundabouts for smooth drive between Greater Noida, Ghaziabad. Retrieved Oct 19, 2020 from https://timesofindia.indiatimes.com/city/noida/5-roundabouts/articleshow/39195913.cms

2018a. Cops to manage traffic real time with Google data. Retrieved Oct 19, 2020 from https://timesofindia.indiatimes.com/city/kolkata/traffic-real-time-with-google-data/articleshow/62466342.cms

2018b. Noida plans one-way traffic in Sector 18. Retrieved Oct 19, 2020 from https://timesofindia.indiatimes.com/city/noida/one-way-traffic/articleshow/65314720.cms

2019a. Bengalureans don't have to wait at traffic signals for too long, thanks to Google Maps data. Retrieved Oct 19, 2020 from https://economictimes.indiatimes.com/magazines/panache/bengalureans-google-maps-data/articleshow/70580654.cms

2019b. Hyderabad traffic dept to use Google Maps' real-time data to manage signals. Retrieved Oct 19, 2020 from https://www.thenewsminute.com/article/hyderabad-traffic-dept-use-google-maps-real-time-data-manage-signals-107797

Raakheshsubhash Arumuga Rajan Raghesh Krishnan K Arunachalam Muthupalaniappan, B Shreehari Nair. 2019. Dynamic Control of Traffic Signals using Traffic Data from Google Maps and Road Cameras. https://www.ijrte.org/wp-content/uploads/papers/v8i2S3/B11270782S319.pdf

Thierry Bouwmans. 2014. Background modeling and Foreground Detection for video surveillance: Traditional and Recent Approaches, Benchmarking and Evaluation. http://www.crcpress.com/product/isbn/9781482205374.

Sachin Chauhan, Kashish Bansal, and Rijurekha Sen. 2020. EcoLight: Intersection Control in Developing Regions Under Extreme Budget and Network Constraints. *Advances in Neural Information Processing Systems 33 pre-proceedings (NeurIPS 2020)*.

Sakitha Kumarage, Saman Bandara, and Dimantha De Silva. 2018. Use of Google Traffic Data in Traffic Analysis. https://doi.org/10.13140/RG.2.2.27486.95041

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. 779–788. https://doi.org/10.1109/CVPR.2016.91

Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. 1290–1298.

Huichu Zhang, Zhenhui Li, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, and Haiming Jin. 2019. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. *The World Wide Web Conference on - WWW '19* (2019). https://doi.org/10.1145/3308558.3314139