ERL: Edge based Reinforcement Learning for optimized urban Traffic light control

Pengyuan Zhou*, Tristan Braud†, Ahmad Alhilal†, Pan Hui*†, Jussi Kangasharju*
*Department of Computer Science, University of Helsinki
†Department of Computer Science and Engineering, The Hong Kong University of Science and Technology

***Abstract*—Traffic congestion is worsening in every major city and brings increasing costs to governments and drivers. Vehicular networks provide the ability to collect more data from vehicles and roadside units, and sense traffic in real time. They represent a promising solution to alleviate traffic jams in urban environments. However, while the collected information is valuable, an efficient solution for better and faster utilization to alleviate congestion has yet to be developed. Current solutions are either based on mathematical models, which do not account for complex traffic scenarios or small-scale machine learning algorithms. In this paper, we propose ERL, a solution based on Edge Computing nodes to collect traffic data. ERL alleviates congestion by providing intelligent optimized traffic light control in real time. Edge servers run fast reinforcement learning algorithms to tune the metrics of the traffic signal control algorithm ran for each intersection. ERL operates within the coverage area of the edge server, and uses aggregated data from neighboring edge servers to provide city-scale congestion control. The evaluation based on real map data shows that our system decreases 48.71% average waiting time and 32.77% trip duration in normally congested areas, with very fast training in ordinary servers.**

I. Introduction

Traffic congestion is continuously rising in most urban areas around the world. Multiple factors contribute to this situation, among which the increasing number of vehicles, the inadequate infrastructure, and the distribution of points of interests around the city. Traffic congestion affects the environment, the individual wellbeing of citizens, and has a considerable impact on the economy. In 2017, drivers spent on average 102 peak hours in congestion in Los Angeles, 91 in Moscow and New York City, 74 in London, and 69 in Paris [1]. Current solutions either rely on traffic center's control or on centralized cloud services such as Google map navigation. However, the continuous increase in urban congestion shows that these solutions need drastic improvement.

Traffic congestion includes a predictable part caused by traffic increase and a non-negligible unpredictable part caused by incidents. Several models already account for the predictable part of congestion. However, to the best of our knowledge, few studies focus on alleviating traffic congestion caused by accidents and incidents. Moreover, although predictable traffic congestion can be solved through long-term infrastructure investments, the consequences of incidents can only be handled in real-time. For these reasons, the best solution to mitigate the impact of traffic congestion is to empower the signal timing plans with traffic-responsive capabilities while notifying incidents and rerouting neighbor vehicles.

In this paper, we propose ERL, an integrated framework to optimize traffic signals and keep drivers updated with the newest traffic conditions. ERL divides the urban traffic into areas defined at intersection, neighborhood and district level. Each level is optimized by a "local edge" device coordinating the edge servers at lower levels. This modular design provides fine-grained traffic optimization while considering the whole city traffic in real time. As the three levels of optimization run in parallel, a local area can optimize its traffic signal plan quickly without waiting for city-scale orchestration. On the other hand, ERL also allows reorganizing the traffic at a larger scale in the case of massive congestion.

ERL's architecture is massively distributed and relies on a pervasive deployment of edge servers, including local monitors at intersections and edge servers at base stations and aggregation points. By focusing on a given small-scale area, ERL allows to considerably simplify the algorithms implemented in each server. For instance, the machine algorithms employed within edge servers are specifically trained for a given area, which limits the possible outcomes. This structure allows for the deployment of specialized lightweight edgelets that require minimal computing capabilities at each level.

The contributions of this paper are as follow: 1) *Design of ERL*, an integrated framework for handling traffic congestion in real-time at city-scale, 2) *Usage of Reinforcement Learning (RL) algorithms to automatically control the traffic signals* depending on the ingoing traffic. To the best of our knowledge, ERL is the first reinforcement learning proposal to optimize traffic signals on neighborhood scale (see section. II), 3) *Extensive simulation* based on real-world data to evaluate the traffic improvement brought by ERL.

The rest of paper is structured as follows. We cover related work in Section II. In Section III we present the overall system design and traffic model. We give practical details of our algorithm in Section IV and present our evaluation in Section V.

II. Related work

Researchers have put a lot of effort into optimizing traffic using data analysis based on auxiliary instruments and techniques [2]. Most studies either adapt traffic lights or encourage drivers to take better decisions to alleviate congestion [3]–[5]. ERL focuses on adapting traffic lights while providing the required fast interactive controls required for rerouting drivers.

Artificial intelligence: Most related works in this area nowadays adopt reinforcement learning algorithm with different adaptations for specific goals. However, the action to take mostly falls in two approaches: turning on/off the light directly [6]–[8] or change the light phase directly [9] based on trained DNN. The obvious disadvantage of the former approach is that immediately transitioning from the current traffic signal phase to the selected action can cause incidents. Though authors in [7] proposed to add additional traffic signal phase configurations preceding the chosen action, different intersections have varied traffic light phase group which require a number of specific configurations. The same issue exists for the latter approach, which also requires additional effort [9].

Moreover, these approaches require a large amount of data to scale to multiple traffic lights,because they have to determine the action for each intersection's lights explicitly. For instance, a typical intersection can easily have more than 6 phases and the training for 15 intersections requires more than 470 GB memory space. Albeit similarly, neighboring lights still have differences in their traffic conditions, preventing coupling to improve the overall performance. As such, local decisions without awareness of neighborhood congestion can hurt traffic control performance by disturbing any inherent city-scale traffic balance through greedy local decisions.

Our proposal, on the other hand, adapt traffic lights indirectly by tuning the thresholds of the phase control algorithm in each intersection. With this design, ERL tunes the **sensitivities** of traffic lights instead of changing them directly. As such, adapting the sensitivities of neighboring traffic lights as a group is reasonable since they experience similar traffic conditions, while leaving each intersection some freedom to decide its light phase depending on its own traffic condition. ERL has much better scalability and faster DNN training.

III. System Design

In this section, we introduce the architecture of ERL and describe the major communication processes. To provide a comprehensive understanding of the system function, we also include the processes of V2E (Vehicle to Edge) which is out of the scope of this work (refer to our previous work [10]).

A. System Architecture

ERL involves around two key layers: the **device layer** and the **edge layer**. The **device layer** includes the vehicles, roadside buildings, infrastructures, traffic signals, intersection monitors, and any other devices involved in the vehicular network. In the rest of this paper, we assume there is at least one device monitoring each intersection. This device embeds video cameras facing all directions and is capable of wireless communication. It therefore captures and transmits all nearby traffic data. We assume the monitor and traffic lights at each intersection are co-located and can transfer data to each other freely and instantly. We call "client" any object in the device layer that transfers data to the ES. The **edge layer** host the ESes at the core of our architecture. We distribute these ESes hierarchically in two tiers. The first tier (T1 ES) consists of ESes co-located with the base stations at the access network level[1]. Second tier ESes (T2 ES) are co-located with aggregation points in the core network. Our placement scheme provide ESes with faster awareness of traffic condition, congestion, and incident, while minimizing the average distance of vehicle to edge and deployment cost. Finally, a remote cloud may provide on-demand backup and aggregation capabilities, which are not our major concern in this work. Please refer to [10] for more details and communication processes.

B. Traffic Model

Due to space limitation, we briefly describe our traffic model. We formulate our model based on a macroscopic model [11] that yields a sufficiently accurate description of the changing traffic flows in large areas (e.g. a central urban area), given the road network, the traffic load and the traffic conditions. Each ES controls the traffic lights in its covered area. To avoid overlapped control demands from different ESes, the assignment of traffic lights to ESes is predefined by the system. Let us assume there are $\mathcal{E}$ T1 ESes in the road network, each of which covers an area consisting of a unidirectional link set $\mathcal{L}^e = \{\mathcal{L}_i | i = 1, ..., I^e\}$ ($e \in \{1, ..., \mathcal{E}\}$). We use e to indicate both the index of ES and its covered area in the rest of this paper. Each link represents a lane connecting two subsequent intersections. Lane changing is not considered.

We assume the traffic lights in the urban area pertain to a common signal timing plan, characterized by a fixed cycle containing $\mathcal{F}$ phases. A phase refers to the time duration of the green lights for a given direction. Let $n_i(k)$, N_i, I_{int} respectively represent the number of vehicles in L_i at the beginning of the kth cycle, the capacity of L_i, and the indices of the internal links inside an ES-covered area. The problem can be formulated as follows:

$$\min_{t^f(k)} N_e(k) = \min_{t^f(k)} \frac{1}{N} \left[\sum_{k=1}^{N} \sum_{i \in I_{int}} n_i(k) \right] \tag{1}$$

for internal ES-covered area traffic optimization, where $t^f(k)$ is the duration of phase fin the kth cycle, $N\_e(k)$ is the total number of vehicles in the area covered by ES e.

IV. Algorithm

The optimization process includes three parallel and interactive threads on three distinct levels, i.e., *intersection* level, *intra ES covered area* level and *inter ES covered area* level, respectively performed by traffic lights, T1 ESes and T2 ESes. The basic optimization of traffic is performed by the individual sets of traffic lights at each intersection. Each traffic light set adapts its red and green light phases to minimize the waiting queue at the intersection, thanks to phase adaptation algorithms. At a higher layer, T1 ESes optimize the traffic in their covered areas, by tuning the metrics of the traffic light control algorithm across the intersections in the area. We base the tuning of metrics on reinforcement learning. T2 ESes

[1]Base station in this paper refers to the entity at the edge of the fixed network, e.g., BTS, eNB, and gNB.

optimize the urban traffic, by tuning the degree of optimization of each T1 ES according to the traffic density.

Intersection level: traffic lights are able to capture the length of jam thanks to their embedded cameras, and adapt the duration of phases based on a traffic light control algorithm. This algorithm relies on the following parameters: the *decision time interval* (time it takes to decide an adaptation) $\mathbf{t_{decide}}$, the *decision threshold* $\mathbf{TH}$ and the *looking distance* [12], [13]. For each intersection, the traffic lights have an initial signal cycle T_c, consisting of the phases of green lights in four directions, respectively t_E, t_e, t_N and $t_n t_E$, t_e, t_N and t_n We consider two *monitoring metrics* on intersection level light adaptation: the average vehicle speed and waiting queue length. M_E, M_e,M_N and M_n indicate the number of vehicles waiting in the aforementioned directions. After t_{decide}, if the ratio of the monitoring metric is larger than the threshold, the system extends the phase of green light for the direction with worse performance. Meanwhile, it decreases the equivalent length of the green phase for the opposite direction. We consider $M_x^{ratio} = (S_y - S_x/S_y)$ to dictate the ratio of average speed in direction x, $M_x^{ratio} = (W_x - W_y/W_x)$ to dictate the ratio of waiting queue length in direction x, where y indicate the opponent direction of x.

Intra area optimization: T1 ESes carry out internal traffic optimization by tuning the metrics of traffic light algorithms at the intersections in their coverage area. For simplicity, we only consider the threshold metric TH of the intersection level algorithm to be tuned. With I2E (infrastructure to edge) communication, T1 ESes collect the monitoring data of their covered area in close to real-time and tune the traffic lights according to Algorithm 1. The optimization uses reinforcement learning algorithm, based on a deep neural network to learn the optimal traffic signal control metric. We adopted deep Q-learning to provide an adaptive algorithm responding to dynamically changing traffic condition. The advantage of Q-learning for traffic signal control is described in more details in a study by Abdulhai et al. [14]. Next, we define the intra area state S_t, agent action A_t and reward R_t.

Intra Area State: ES $\mathbf{e}$ needs the following intra area information to tune the metrics of traffic signal control algorithm: traffic flow, average vehicle speed at each road and signal control algorithm state. To represent the traffic flow and the vehicle speed, we collect the number of vehicles in each lane and their average speed into a matrix of traffic flow $\mathbf{F_e}$ and a matrix of average vehicle speed $\overline{\mathbf{V}}_\mathbf{e}$. The number of vehicles in each lane is normalized by lane capacity and recorded at the corresponding entry of matrix $\overline{\mathbf{V}}_\mathbf{e}$. In the following equations, $A^\intercal$ represents the transposition of matrix A. The matrix $\mathbf{F_e}$ of traffic flow for all roads in the area is defined as follows:

$$F = [n_1(t), n_2(t), \ldots, n_{I^e}(t)]^\intercal \tag{2}$$

where $n_i(t)$ is the number of vehicles in lane L_i at time t, I^e is the number of lanes in the area covered by e. Similarly, the matrix of average speed for all roads in the area is:

$$V = [\overline{v}_1(t), \overline{v}_2(t), \ldots, \overline{v}_{I^e}(t)]^\intercal \tag{3}$$

We use a vector $\mathbb{TH}$ to represent the threshold metrics of the signal control algorithm of each intersection.

$$TH = [\overline{th}_1(t), \overline{th}_2(t), \ldots, \overline{th}_{I^e}(t)]^\intercal \tag{4}$$

where $n_i(t)$ is the number of vehicles in lane L_i at time t, l^e is the number of traffic lights in the area covered by e. At the beginning of time step t, the agent observes intra area state $S_t = (\mathbb{F}, \mathbb{V}, \mathbb{TH})$ for intra area traffic control.

Agent Action: After observing intra area state S_t at the beginning of time step t, the agent chooses one action $A_t \in \{-1, 0, 1\}$: decreasing, maintaining or increasing the value of threshold metric th for each traffic light. The threshold TH can be seen as the sensitivity of the light control algorithm: the algorithm changes the green light phase more frequently with a lower threshold. For instance, in a central area, the traffic flow varies a lot from morning to evening, which requires a higher sensitivity to adjust the signals. Rural areas, on the other hand, can have small traffic flow all the time, which can be satisfied with "retarded" light control.

Reward: The reward is the change of the speed and number of vehicles between two neighboring cycles as follows:

$$R_t = A * (V_{t+1} - V_t) + B * (N_t - N_{t+1}) \tag{5}$$

where A and B are weight metrics.

Agent Goal: The overall goal of ERL is to optimize the traffic control. As a result, ERL makes the traffic flow smoother, with shorter waiting times and higher average speeds in the long run. The agent needs to find an action policy $\pi*$ that maximizes the cumulative future reward as follows (Q-value):

$$Q^\pi(s, a) = \exp\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi\right] \tag{6}$$

where γ is a discount parameter, $0 \leq \gamma \leq 1$, reflecting the weight the agent puts on future rewards. To put it another way, the goal is to find following $\pi*$:

$$\pi* = \arg\max_\pi Q_\pi(s, a) \tag{7}$$

Deep Neural Network (DNN): We use a simplified neural network architecture. The network takes the state vector S_t as the input, following two hidden fully connected layers of 2000 and then 3000 neurons with rectifier nonlinear activation functions. The output layer is $|A_t|^{l^e}$ (l^e is the number of traffic lights) neurons with linear activation functions.

To decrease the output layer dimension, we propose to *combine the threshold adaptation of neighbor traffic lights. Since our agent action is to tune the threshold value of the intersection level algorithm, combining the threshold adaptation of neighbor traffic lights does not couple their phase adaptations.* To put it in a simpler way, we only couple the sensitivity of neighbor traffic lights. Their phase changes depend on the real traffic. We train the network using ADAptive Moment estimation (Adam) [15] which fits for fast convergence and satisfactory performance [16]. We adopt the ϵ-greedy method to let the agent selects the action with the current biggest estimated Q-value with probability $1 - \epsilon$ and randomly selects

Algorithm 1 Intra ES area algorithm

Initialize DNN network with random weights θ;
Initialize ϵ,γ,N;
for epoch $= 1$ to N **do**
  Initialize intra ES area state S_1;
  Initialize action A_0;
  Start new time step;
  for time $= 1$ to T seconds **do**
    Based on observed state S_t,
    the agent selects action $A_t = \arg\max_a Q(S_t, a; \theta)$ with probability $1 - \epsilon$ and randomly selects an action A_t with probability ϵ;
    if $A_t == A_{t-1}$ **then**
      keep thresholds unchanged
    else
      send new thresholds to the traffic lights in covered intersections according to A_t
    end if
    Increment simulation by period t
    The agent observes reward R_t and next state S_{t+1};
    Store observed experience (S_t, A_t, R_t, S_{t+1}) into replay memory M;
  end for
  Randomly draw minibatch samples (S_i, A_i, R_i, S_{i+1}) from M
  Batch training: input state and targets and train the network according to Eq. 6 and Eq. 7.
end for

one action with ϵ at each time period (see Algorithm 1). To reduce cost, we adopt minibatch method. The agent randomly draws a batch of samples from the replay memory M to form input data and target pairs and update DNN weights θ by Adam algorithm. In section. V, we test different action periods, each of which corresponds to a feasible minibatch size.

Inter area optimization: Each ES coverage area can be considered as a "big intersection" in which the internal traffic flow is optimized with *intra area optimization*. Since each ES is placed in the cluster center based on traffic density, the traffic flow between the ES coverage areas is more likely to be lower. However, if ES areas are overlapping, the optimization of a congested area may affect the other areas. That is to say, if an ES relieves the congestion inside its covered area, output links will have higher traffic flow towards neighboring areas. These areas will then see their internal traffic flow increase. For this reason, Tier 2 ESes optimize the inter area traffic following the same methodology with the intersection level algorithm, that is, slowing down the optimization of a T1 ES area if its impact on neighboring areas outdoes the improvement of its internal traffic. In this paper, we focus on evaluating the first two optimizations, and leave the evaluation for inter area optimization for future work.

The motivation behind this design is threefold: (i) Light control on intersection level should be in real time, that is, phase decision should be made by a simple algorithm. Threshold-based algorithms fulfills this requirement. (ii) Though being fast, threshold-based algorithms lack the ability to optimize traffic light control facing different traffic conditions and surrounded road maps. The reinforcement learning algorithm tunes the thresholds in different intersections to provide comprehensive optimization. (iii)The distribution of ERL's computational complexity fits the nature of infrastructure deployment, which is, intersection light with limited resource works better with a simpler algorithm, which edge server is capable of much heavier computation. Yet, for fast computation and due to resource upper bound, it is reasonable to assign local traffic light control to an edge server instead of a larger scale task.

V. Evaluation

We deployed a ES on a Linux server, with a single core Intel(R) Xeon(R) CPU E5-2680 v4, 10GB of memory partition, and an NVIDIA Tesla P100 GPU. The hardware specification of our ES is similar to a cheap-priced edge server in 2018 [17]. As such, we test the system performance without replying on expensive hardware. We build our reinforcement learning algorithm based Keras [18] and run the tests on the Simulation of Urban MObility (SUMO) [19] simulator.

A. Simulation Settings

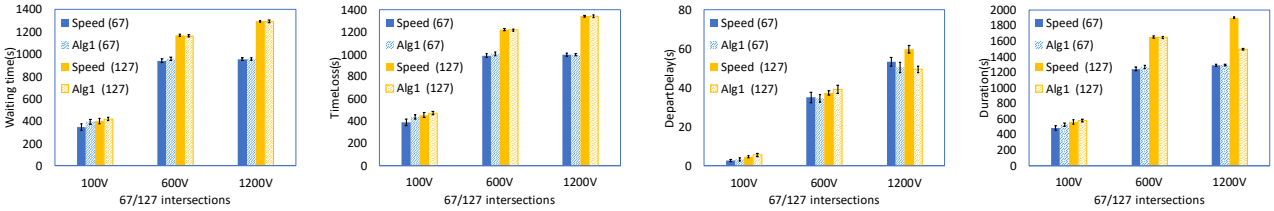
To extensively test our system, we focus on different metrics that may influence the performance. The overall goal of simulation is to compare the system performance in different scenarios, find out the best metrics, and the suitable scenarios.

Map: We evaluate our system on real map data extracted from Open Street Map (OSM) [20]. We select different scales centering on Times Square, New York City, which is one of the most congested area in the world. This area contains dense distribution of road and traffic lights, which fits our goal of testing the system on traffic light control. Moreover, it gives a lower bound of system performance with a large proportion of unidirectional roads and not-four-direction traffic lights. We test on two areas containing 67 and 127 traffic lights.

Traffic: We generate the traffic with a random algorithm, that a specific amount number of vehicles are generated per hour/lane-kilometer, each of which has a random departure and destination road. We test on 100, 600 and 1200 vehicles per hour/lane-kilometer. Taking 100V (vehicle) as an example, it means that 400 vehicles will be generated on a 4-lane unidirectional road per hour. Up-to-date public New York traffic data is out of reach. Therefore, to map the generated traffic to real life, we compare it with the traffic data in London central area[2]. We find that $100V/h/lane-km$ is comparable with the peak time traffic in central London (only with similar size of roads). Meanwhile, $600V/h/lane-km$ and $1200V/h/lane-km$ are paranormal traffic volume which we only use to test the system performance in extreme scenarios.

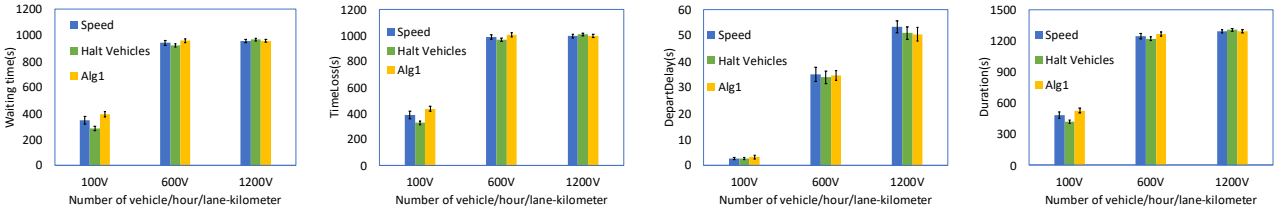
Monitoring metric: As introduced in section. IV, each intersection can tune its traffic lights using the intersection

[2]https://data.gov.uk/dataset/gb-road-traffic-counts



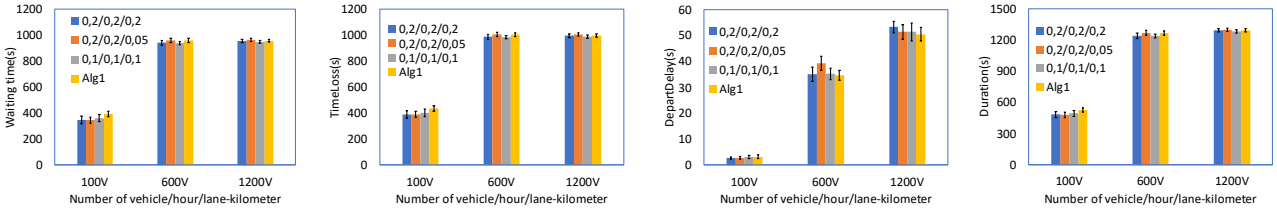
(a) Average waiting time. (b) Average time loss. (c) Average depart delay. (d) Average trip duration.

Fig. 1: Statistics on different scales



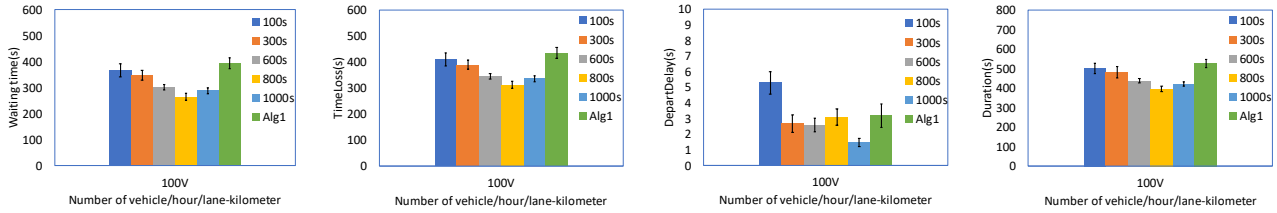
(a) Average waiting time. (b) Average time loss. (c) Average depart delay. (d) Average trip duration.

Fig. 2: Statistics on different monitors



(a) Average waiting time. (b) Average time loss. (c) Average depart delay. (d) Average trip duration.

Fig. 3: Statistics on different units



(a) Average waiting time. (b) Average time loss. (c) Average depart delay. (d) Average trip duration.

Fig. 4: Statistics on different periods

level algorithm based on average vehicle speed or waiting queue length. To extend the measurement to cover the whole roads, we use number of vehicles with speed of less than $0.1m/s$ $(halting)$ on each lane instead of waiting queue length. We test and compare the two metrics in our evaluation.

Adaption period: As mentioned in section. IV, we test different adaptation period, i.e., $100s$, $300s$, $600s$, $800s$ and $1000s$ to find out the best period for adapting the threshold of the intersection level algorithm. We set the test interval as such so the lights phases can be tuned upon a appropriate period.

Adaption unit: We also test different adaptation unit, i.e., how much a change being made to a threshold value in each action. Unlike other works, we do not assume a specific threshold works best for all scenarios. Here we test on different adaptation units of *default value/increment/decrement*, i.e., $0.2/0.2/0.2$, $0.2/0.2/0.05$ and $0.1/0.1/0.1$. To be noted, authors in [13] use 0.1 as the default threshold value.

B. Simulation Results

We evaluate the performance based on following metrics: average waiting time (time in which the vehicle speed was below $0.1m/s$), average time loss (time lost due to driving below the ideal speed), average depart delay (time the vehicle had to wait before it could start his journey due to lack of road

space), and average trip duration. Unless stated differently, all of our test are with one hour length and we use the approach with adaptation period: *300s*, adaptation unit:$0.2/0.2/0.2$, monitoring metric: *average speed* by **default**. And in all the tests, we use *the intersection level algorithm* (without Algorithm. 1) as a comparison approach.

First, Figure. 1 to Figure. 4 all show that our system improve all the performance metrics in all the scenarios of $100V/h/lane - km$. On the other hand, in the two extreme scenarios of 600 and 1200 $V/h/lane - km$, our system provides very limited help, even minor controversial impact in some cases (Figure. 1d). This shows that our system can alleviate traffic congestion in normal scenarios. However, in extremely bad scenarios (600V and 1200V/h/lane-km), the traffic is so congested that there is nothing much we can do to help. For instance, the average waiting time reaches 1200s in 127-traffic-light scenarios (Figure. 1a), which is about 75% of the average trip duration. This is obviously beyond the scope that traffic light control can help with.

Next, Figure. 1 shows that ERL can provide considerable improvement in 67-light map. In 127-light map, the improvement becomes much smaller. This shows that our work fits small scale maps, in accordance with our expectation. Because we wanted to realize fast control feedback in distributed edge servers, each of which covers a small area. Only within small area, the amount of collected data allows fast DNN training and tuning. Cloud collects data at a larger scale, but provides much slower DNN training as a result. As such, cloud service certainly can not provide DNN training and control feedback as fast as ERL. Figure. 2 shows that monitoring the number of halting vehicles provides better performance than average speed. Figure. 3 and Figure. 4 show that approach unit $0.2/0.2/0.2$ and period 800s outperform others.

As a summary, ERL trains DNN based on **one-hour** traffic data within **9m30s** on average. Therefore within every 10 minutes, ERL is able to update the DNN based on previous hourly traffic data. The improvement in our evaluation includes **decreasing at most** 48.71% **average waiting time,** 39.49% **time loss,** 3.12% **depart delay and** 32.77% **trip duration (Figure. 4), on the map of central New York area that contains 67 traffic lights with normal congestion level (100V/h/lane-km)**.

VI. Conclusion

In this work, we present ERL, an architectural framework for traffic lights optimization. Our system exploits the low latency of Edge servers to provide fast DNN training and control feedback. Thanks to its layered architecture and algorithm, ERL runs optimization at intersection, neighborhood and city level that allow for different fine-grained and scale of traffic control. As a first step, we evaluate the first two layers of optimization in this work. Requiring only ordinary hardware, ERL can decrease 48.71% average waiting time at normal congestion scenario. Unlike other works, we propose the architectural algorithm and select threshold of phase control as the action target. With this indirect control methodology, we enable the coupling of neighboring lights adaptation and decrease the dimension of action space. allowing ERL to scale to city block size with fast training and control feedback.

As a first step, we explore the performance of the algorithm on intra ES area level. In the future work, we will focus on improving the intra ES area algorithm and evaluate inter ES traffic optimization.

References

[1] G. Cookson and B. Pishue, "Inrix global traffic scorecard," *INRIX Research, February*, 2017.

[2] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen *et al.*, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang *et al.*, "Traffic flow prediction with big data: A deep learning approach." *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[4] R. L. Bertini, S. Hansen, A. Byrd, and T. Yin, "Experience implementing a user service for archived intelligent transportation systems data," *Transportation research record*, vol. 1917, no. 1, pp. 90–99, 2005.

[5] Y. Ding, C. Chen, S. Zhang, B. Guo, Z. Yu, and Y. Wang, "Greenplanner: Planning personalized fuel-efficient driving routes using multi-sourced urban data," in *Pervasive Computing and Communications (PerCom), 2017 IEEE International Conference on*. IEEE, 2017, pp. 207–216.

[6] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," *arXiv preprint arXiv:1705.02755*, 2017.

[7] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[8] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[9] X. Liang, X. Du, G. Wang, and Z. Han, "Deep reinforcement learning for traffic light control in vehicular networks," *arXiv preprint arXiv:1803.11115*, 2018.

[10] P. Zhou, W. Zhang, T. Braud, P. Hui, and J. Kangasharju, "Arve: Augmented reality applications in vehicle to edge networks," in *Proceedings of the 2018 Workshop on Mobile Edge Communications*. ACM, 2018, pp. 25–30.

[11] A. Barisone and D. Giglio, "A macroscopic traffic model for real-time optimization of signalized urban areas," *41st IEEE Conference on Decision and Control*, pp. 900–903, 2002.

[12] D. Krajzewicz, E. Brockfeld, J. Mikat, J. Ringel, C. Rossel, W. Tuchscheerer, P. Wagner, and R. Wosler, "Simulation of modern traffic lights control systems using the open source traffic simulation SUMO," *In Proceedings of the 3rd Industrial Simulation Conference, Berlin, Germany*, pp. 229–302, 2005.

[13] E. Brockfeld and P. Wagner, "Agent Based Traffic Signals Regulating Flow on a Basic Grid."

[14] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[17] DELL, "Poweredge c4130 rack server optimized for gpus and co-processors," 2018. [Online]. Available: https://www.dell.com/en-us/work/shop/povw/poweredge-c4130

[18] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[19] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[20] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Ieee Pervas Comput*, vol. 7, no. 4, pp. 12–18, 2008.