TinyLight: Adaptive Traffic Signal Control on Devices with Extremely Limited Resources

Dong Xing[1] , **Qian Zheng**[1] , **Qianhui Liu**[1,2] and **Gang Pan**[1,*]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]Department of Electrical and Computer Engineering, National University of Singapore, Singapore
{dongxing, qianzheng}@zju.edu.cn, qhliu@nus.edu.sg, gpan@zju.edu.cn

Abstract

Recent advances in deep reinforcement learning (DRL) have largely promoted the performance of adaptive traffic signal control (ATSC). Nevertheless, regarding the implementation, most works are cumbersome in terms of storage and computation. This hinders their deployment on scenarios where resources are limited. In this work, we propose TinyLight, the first DRL-based ATSC model that is designed for devices with extremely limited resources. TinyLight first constructs a super-graph to associate a rich set of candidate features with a group of light-weighted network blocks. Then, to diminish the model's resource consumption, we ablate edges in the super-graph automatically with a novel entropy-minimized objective function. This enables TinyLight to work on a standalone microcontroller with merely 2KB RAM and 32KB ROM. We evaluate TinyLight on multiple road networks with real-world traffic demands. Experiments show that even with extremely limited resources, TinyLight still achieves competitive performance. The source code and appendix of this work can be found at https://bit.ly/38hH8t8.

1 Introduction

The latest progress of adaptive traffic signal control (ATSC) presents two different situations from views of research and practice. From the view of research, many studies on ATSC have shown superior performance over traditional fixed-cycle solutions [Wei *et al.*, 2020]. Especially, with the development of deep reinforcement learning (DRL) [Xing *et al.*, 2021], works on DRL-based ATSC exhibit great potential when applying them to intersections with various settings. From the view of practice, however, the deployment rate of ATSC in reality is still far from satisfactory. In 2019, less than 5% of signalized intersections in the USA adopt ATSC systems [Tang *et al.*, 2019]. In developing regions, the application of ATSC is more uncommon [Chauhan *et al.*, 2020]. As traffic signal control is a field with broad application background, how to turn research outcomes into practice is critical.

One key factor impeding the application of ATSC is its high expense. For example, the cost of ATSC in Florida ranges from $30,000 to $96,400 per intersection [Elefteriadou *et al.*, 2019]. Recent studies on tiny machine learning enable simple algorithms such as kNN to work on cheap microcontroller units (MCUs) [Branco *et al.*, 2019]. This inspires us to adopt such devices to partially reduce the cost of ATSC.[1] An MCU encapsulates processors, memory and I/O peripherals into a single chip. Comparing with a general workstation, the resources on MCUs are much limited (*e.g.*, most MCUs have only KB of memory and mediocre clock rate). This makes its price relatively low (less than $5 in our case) and therefore affordable in many practical scenarios. Nevertheless, its limitation on resource also poses new constraints on the storage and computation of ATSC implementations.

By interacting with the environment to adapt to real-time traffic flows, recent DRL-based solutions achieve superior performance over traditional ones. However, when considering their deployment, most works do not take into account the issue of resource consumption, leaving their performance on MCUs (or other embedding devices) undetermined. From the storage aspect, many works rely on complex network structures to maintain the representation power, and their memory requirement often exceeds the capacity of a single MCU. From the computation aspect, it is common for existing works to involve massive parallel operations, which greatly affect their response time if GPU is not available. These factors are practical during the deployment. However, we observe that they were rarely discussed in prior works.

To address these issues, we propose TinyLight, the first DRL-based ATSC model that is designed for devices with extremely limited resources. Specifically, TinyLight first constructs a super-graph to associate a rich set of candidate features with a group of light-weighted network blocks. This super-graph covers a wide range of sub-graphs that are compatible with resource-limited devices. Then, we ablate edges in the super-graph automatically with a novel entropy-minimized objective function, which greatly diminishes the resource consumption of TinyLight. In our experiments, the model we obtained takes only 4KB ROM. However, it still reaches competitive performance within 0.1s response time

*Gang Pan is the corresponding author.

[1]We choose MCU as it is a representative device with extremely limited resources, but our work is *not* restricted to this device.

on devices with extremely low clock rate (8MHz).

As a demonstrative prototype of our model's deployment, we implement TinyLight on a standalone ATmega328P – an MCU with merely 2KB RAM and 32KB ROM. This MCU is readily available on the market and costs less than $5, making our work applicable in scenarios with low budgets. We evaluate TinyLight on multiple road networks with real-world traffic flows. Experiments show that even with extremely limited resources, TinyLight still achieves competitive performance. In this way, our work contributes to the transformation of ATSC research into practice. Although we focus on the task of traffic signal control, our methodology can be generalized to many fields that are also cost sensitive [Liu *et al.*, 2020].

2 Related Works

This section discusses related works of ATSC from both perspectives of research (works with no hardware involved) and practice (works with hardware involved).

2.1 ATSC Research

Many pioneering works on ATSC are manually designed by experts in transportation engineering. For example, SOTL [Cools *et al.*, 2008] counts vehicles with or without the right of way and compares them with empirical thresholds. Max-Pressure [Varaiya, 2013] assumes downstream lanes have infinite capacity and greedily picks phase that maximizes the intersection's throughput. These works highly depend on human experience in transportation systems, making them too rigid for traffic flows that change dynamically.

Instead, DRL-based solutions directly learn from traffic flows and adapt to specific intersections more flexibly. To determine the next phase for a single intersection, [Zheng *et al.*, 2019] compares phases in pairs. [Oroojlooy *et al.*, 2020] applies attention on both lanes and phases, and [Jiang *et al.*, 2021] extends this idea to dynamic lanes. For scenarios with multiple intersections, [Wei *et al.*, 2019b] and [Yu *et al.*, 2020] incorporate the influences of neighboring intersections with graph networks. [Rizzo *et al.*, 2019] develops time critic policy gradient methods for congested networks. [Chen *et al.*, 2020] uses pressure to coordinate signals in region-level. Armed with judiciously designed networks, these models achieve superior performance over traditional ones. However, most of them are cumbersome in storage or computation, which impedes their deployment. In contrast, TinyLight is implemented with moderate scales of storage and computation. This enables our model to work on devices with limited resources, making it more practical for the deployment.

2.2 ATSC Practice

A number of commercial ATSC systems [Hunt *et al.*, 1981] have been adopted in modern cities. However, the high costs of these systems constrain their popularity in most regions with tight budgets. Several works manage to narrow the gap between simulation and reality. They achieve this by designing communication platforms to evaluate the performance of general ATSC models in real-world environments [Abdelgawad *et al.*, 2015]. However, these works do not change the implementation of ATSC models, leaving the cost of deployment still expensive. EcoLight [Chauhan *et al.*, 2020] is a threshold-based model for scenarios with primitive conditions. The authors first design a light-weighted network and then approximate it with a set of empirical thresholds to simplify the deployment. However, this process relies heavily on human expertise in the local traffic conditions, making it hard to generalize to other intersections. In comparison, the formation of TinyLight is fully data-driven, which greatly mitigates the model's dependency on manual operations.

Our work has also been influenced by recent works on neural architecture search [Elsken *et al.*, 2019]. Nevertheless, existing studies in this field usually do not position the ultimate model size in the first place. This constraint, however, is the main focus of TinyLight. Unlike previous works, our design for the super-graph only involves light-weighted network blocks, and our proposed entropy-minimized objective function proactively ablates edges in the super-graph to a great extent. This makes our final model extremely tiny and therefore can even work on devices with only KB of memory.

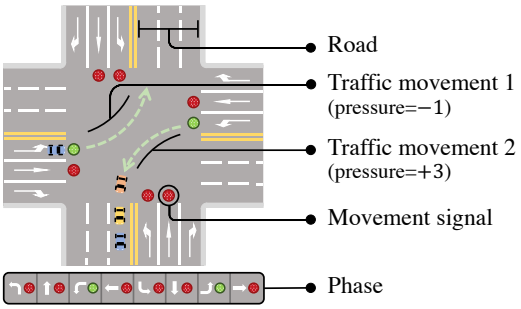


Figure 1: An illustration of the ATSC environment.

3 Preliminaries

This section provides the definition of related terms and a formal description of the ATSC problem setting.

3.1 Term Definitions

We use a four-way intersection (depicted in Figure 1) as an example to introduce the following terms. A *road* is a wide way where vehicles can drive. A *traffic movement* refers to a vehicle flow that starts from an incoming road and ends at an outgoing road. A *movement signal* determines if a traffic movement has the right of way to pass through. A *phase* is a combination of movement signals that do not conflict with each other. The *pressure* measures the net difference between vehicles on the incoming and outgoing roads of a traffic movement. These definitions are consistent with previous works [Wei *et al.*, 2020] and can be naturally extended to other intersection structures.

3.2 Problem Setting

Our problem follows the setting of existing DRL-based works [Wei *et al.*, 2020]. At time step t, the model collects its state $\mathbf{s}_t$ and reward r_t from the environment, based on which it determines a phase a_t from the collection of $|P|$ valid phases to allow some traffic movements to pass through. The chosen phase is executed after 3 seconds of yellow signal to let drivers have enough time to prepare. The target of our

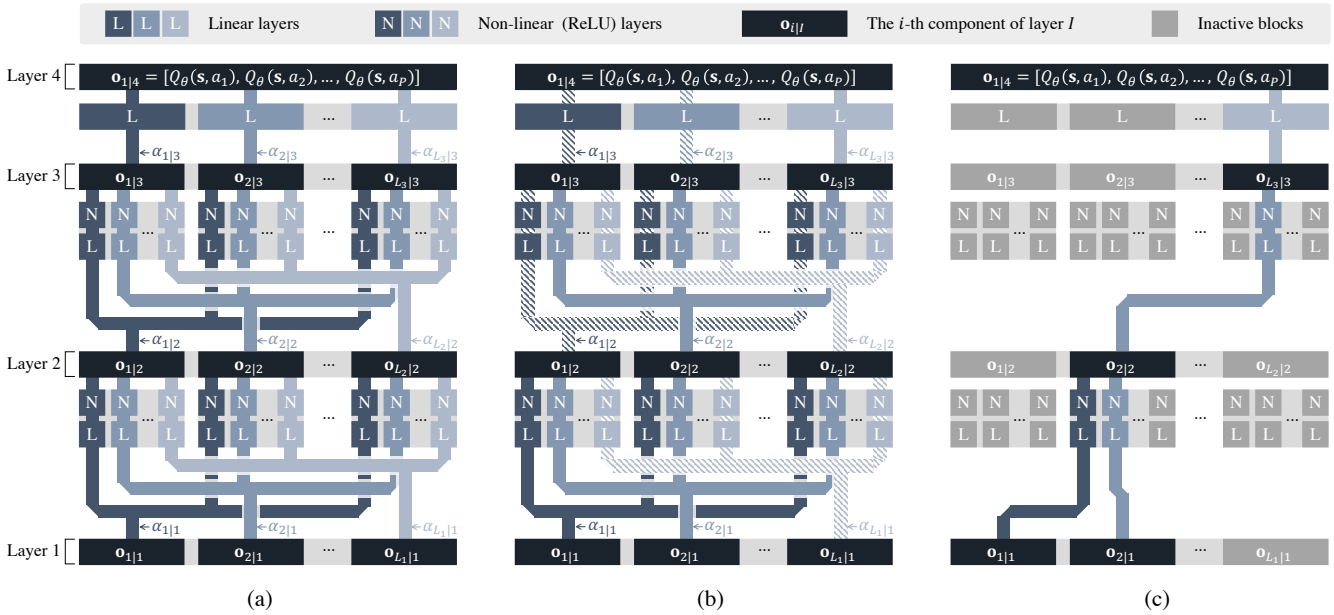


Figure 2: The formation process of TinyLight's final model. (a) The super-graph; (b) The intermediate result during edge ablating (stripped line indicates that the value of corresponding α is relatively smaller); (c) The sub-graph (*i.e.* the policy network).

problem is to maximize the cumulative discounted rewards $\mathbb{E}[\sum_t \gamma^t r_t]$, where γ is the discount factor of future return. For the implementation, we choose negative pressure over intersection at time t as the instant reward r_t, which suggests that intersections with lower pressures are more favored. This reward is correlated with long term performance indexes such as average travel time and throughput [Wei *et al.*, 2019a].

4 Method

This section presents the details of our model. We first describe the implementation of TinyLight in Section 4.1, and then discuss its resource consumption in Section 4.2.

4.1 The TinyLight Model

The Super-Graph

Recall that our goal is to automatically extract a tiny but well-performed sub-graph from a super-graph, making it applicable on devices with extremely limited resources. Therefore, we expect the super-graph should cover a large quantity of tiny sub-graphs that have miscellaneous representation emphases. To meet this requirement, we implement a multi-scale feature extractor to represent a rich set of candidate features. This procedure depicts the road network from various scales, including (incoming and outgoing) lanes, roads, traffic movements, phases and the intersection. On each scale, a variety of traffic statistics are collected, such as the number of vehicles, the sum of their waiting time and so on. In consequence, nearly 40 types of candidate features are prepared for the upcoming procedure of sub-graph extraction, which constitutes a comprehensive representation for the road network.[2]

[2]Appendix A provides the full list of our covered features.

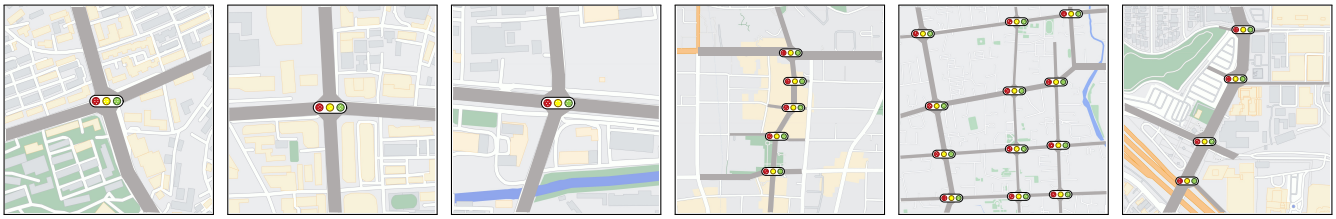
To further exploit these features, we construct a four-layer network structure, depicted in Figure 2a. The 1st layer is reserved for our candidate features. The 2nd and 3rd layers extract high-level representations. The last layer produces the phase. Each layer I has L_I parallel components (denoted as black rectangles) whose output dimensions differ from each other. During sub-graph extraction, only components that are most suitable for the current intersection are remained. For layer $J \in \{2, 3, 4\}$, its j-th component accumulates the transformation of all components from its previous layer I. Every single transformation involves a linear function and a non-linear (ReLU) function, which are light-weighted and can be implemented on resource-limited devices (c.f. Section 5.3). It should be mentioned that the structure of our super-graph is not fixed and can be easily enhanced with deeper layers or stronger network blocks if the resource is abundant.

The Sub-Graph

To enable sub-graph extraction, we let all transformations that start from a common component (e.g., the i-th component of layer I where $I \in \{1, 2, 3\}$) be weighted by a shared non-negative parameter $\alpha_{i|I}$. On each layer I, the sum of all $\alpha_{i|I}$ equals to 1 (ensured by a softmax function). We denote the output of j-th component from layer J as $\mathbf{o}_{j|J}$. Its relationship with $\mathbf{o}_{i|I}$ from the previous layer I is represented as:

$$\mathbf{o}_{j|J} = \sum_i \alpha_{i|I} \cdot \mathrm{ReLU}\left(\mathrm{Linear}_\theta\left(\mathbf{o}_{i|I}\right)\right)$$

where θ denotes parameters of linear function. After extraction, most α will be set to 0, and their corresponding edges will be ablated. Since we restrict the sum of $\alpha_{i|I}$ to be 1 on each layer, there is at least one path that traverses from input to output layer, which guarantees the existence of our final sub-graph (proved in Appendix B.2). In addition, this design



(a) Hangzhou-1 (b) Hangzhou-2 (c) Hangzhou-3 (d) Atlanta (e) Jinan (f) Los Angeles

Figure 3: The road networks for evaluation. (a) Baochu Rd. and Tiyuchang Rd., Hangzhou; (b) Qingchun Rd. and Yan'an Rd., Hangzhou; (c) Tianmushan Rd. and Xueyuan Rd., Hangzhou; (d) Peachtree St., Atlanta; (e) Dongfeng Dist., Jinan; (f) Lankershim Blvd., Los Angeles.

enables many sub-graphs to co-exist in the same super-graph. Even if the number of remaining connections on each layer is restricted to one, the proposed super-graph still covers a total amount of $\prod_I |L_I|$ sub-graphs.

Two Parallel Objectives

Given the above settings, there are two groups of parameters to be optimized. The first is θ (parameters of all linear functions), which determines the performance of our sub-graph. The second is α (weights of all edges), which controls the model's sparsity. As our goal is to obtain a model that is both (i) well-performed and (ii) tiny in size, it can be naturally represented as a multi-objective optimization problem.

The first objective is to improve the performance of our sub-graph. This is made possible by minimizing:

$$\mathcal{L}^{(1)}_{\theta,\alpha} = \mathbb{E}\left[\left(r_t + \gamma \max_{a_{t'}} Q_{\theta,\alpha}\left(\mathbf{s}_{t'}, a_{t'}\right) - Q_{\theta,\alpha}\left(\mathbf{s}_t, a_t\right)\right)^2\right]$$

which is the objective of value-based DRL under the structure defined by α. Following convention, we use $Q_{\theta,\alpha}(\mathbf{s}_t, a_t)$ to denote the expected return when picking phase a_t at state $\mathbf{s}_t$. It is an alias for the a_t-th dimension of $\mathbf{o}_{1|I=4}$. By this objective, both θ and α are optimized to maximize the Q-value, which corresponds to models with better performance.

The second objective is to ablate edges in the super-graph proactively to diminish the size of our ultimate sub-graph. An ideal choice is to minimize the $\mathcal{L}_0$-norm of α. However, this target is unfortunately intractable. To facilitate the optimization, we propose to minimize the sum of entropy of $\alpha_{i|I}$ on each layer I, which is represented as:

$$\mathcal{L}^{(2)}_{\alpha} = \sum_I \mathcal{H}\left(\alpha_{\cdot|I}\right) = \sum_I \sum_i -\alpha_{i|I} \cdot \log\left(\alpha_{i|I}\right)$$

Theoretically, $\mathcal{L}^{(2)}_{\alpha}$ is minimized when there is only one instance of α remained on each layer (proved in Appendix B.3). This makes the final size of our sub-graph controllable, even when the super-graph is designed to be enormously large. Comparing with $\mathcal{L}_0$-norm of α, this objective is differentiable and therefore enables most gradient-based optimizers.

The Optimization

By combining the aforementioned two parallel objectives with one Lagrangian function, we can reformulate our goal as the following entropy-minimized objective function:

$$\mathcal{L}_{\theta,\alpha} = \mathcal{L}^{(1)}_{\theta,\alpha} + \beta \mathcal{L}^{(2)}_{\alpha}, \quad \text{s.t.} \quad \beta \geq 0$$

where β is a non-negative Lagrangian multiplier. We perform the parameter updating with dual gradient descent, which alternates between the optimization of θ and α. In this process, the α proactively ablates connections in the super-graph (Figure 2b), and the θ adjusts behavior of our sub-graph given its current structure. At the end of optimization, we remove most edges with negligible weights and eliminate all network blocks that are not in the path to the output layer. We retain two features in layer 1 for a fair comparison with our comparing baselines. Nevertheless, to manifest that our model is tiny in size, we retain only one component in both layer 2 and 3. This determines the ultimate structure of our sub-graph, which is denoted in Figure 2c. Theoretically, its representation power is guaranteed by the universal approximation theorem [Hornik *et al.*, 1989]. Empirically, we observe that its performance remains competitive even when comparing with many strong baselines.[3]

4.2 Resource Consumption

In addition to the sparse connection of our sub-graph, there are several additional designs to further reduce its consumption on resources. First, although we adopt massive network blocks to implement the super-graph, all the basic operations are restricted to be in the set of {linear, ReLU, summation}. This enables us to implement all modules on embedding devices with no dependency on third-party libraries such as `gemmlowp` / `ruy` (required in TensorFlow Lite) or `QNNPACK` (required in PyTorch), which consume additional memory spaces. Second, we deliberately avoid using complex operators (such as convolution) that cause intensive computations in single-threaded programs. Instead, all our network blocks are chosen with moderate sizes of parameter and computation. This decreases our model's dependency on specialized devices such as GPU. We will quantitatively measure TinyLight's resource consumption in Section 5.3.

5 Experiments

This section presents experimental studies. Section 5.1 introduces related settings. Then, we evaluate TinyLight from both aspects of traffic efficiency in Section 5.2 and resource consumption in Section 5.3, which correspond to our two parallel objectives. We also provide an ablation study for the

[3] Appendix B provides more detailed settings for training.

	Hangzhou-1		Hangzhou-2		Hangzhou-3	
	Travel Time (sec./veh.)	Throughput (veh./min.)	Travel Time (sec./veh.)	Throughput (veh./min.)	Travel Time (sec./veh.)	Throughput (veh./min.)
EcoLight	$196.82_{\pm 18.69}$	$29.73_{\pm 0.54}$	$135.33_{\pm 2.33}$	$23.16_{\pm 0.09}$	$100.45_{\pm 2.42}$	$28.67_{\pm 0.07}$
FixedTime	$282.68_{\pm 2.01}$	$27.59_{\pm 0.04}$	$135.89_{\pm 1.94}$	$23.23_{\pm 0.03}$	$418.05_{\pm 1.85}$	$23.54_{\pm 0.03}$
MaxPressure	$121.23_{\pm 3.07}$	$31.19_{\pm 0.09}$	$138.72_{\pm 1.75}$	$23.10_{\pm 0.05}$	$103.98_{\pm 2.41}$	$28.62_{\pm 0.05}$
SOTL	$250.58_{\pm 3.66}$	$28.69_{\pm 0.06}$	$136.74_{\pm 2.02}$	$23.21_{\pm 0.05}$	$143.35_{\pm 3.55}$	$28.32_{\pm 0.04}$
CoLight	-	-	-	-	-	-
FRAP	$129.65_{\pm 45.77}$	$30.79_{\pm 1.21}$	$137.02_{\pm 34.42}$	$23.09_{\pm 0.37}$	$107.74_{\pm 74.03}$	$28.35_{\pm 1.20}$
MPLight	$128.61_{\pm 62.28}$	$30.81_{\pm 1.16}$	$121.87_{\pm 1.28}$	$23.24_{\pm 0.06}$	$82.48_{\pm 1.26}$	$28.73_{\pm 0.05}$
TLRP	$152.88_{\pm 89.03}$	$30.54_{\pm 1.38}$	$126.70_{\pm 11.30}$	$23.20_{\pm 0.15}$	$83.98_{\pm 6.61}$	$28.72_{\pm 0.07}$
TinyLight	$\mathbf{102.87}_{\pm 2.98}$	$\mathbf{31.36}_{\pm 0.05}$	$\mathbf{121.00}_{\pm 0.85}$	$\mathbf{23.25}_{\pm 0.04}$	$\mathbf{81.79}_{\pm 1.10}$	$\mathbf{28.74}_{\pm 0.07}$

	Atlanta		Jinan		Los Angeles	
	Travel Time (sec./veh.)	Throughput (veh./min.)	Travel Time (sec./veh.)	Throughput (veh./min.)	Travel Time (sec./veh.)	Throughput (veh./min.)
EcoLight	$303.07_{\pm 14.93}$	$47.63_{\pm 5.26}$	$384.43_{\pm 9.64}$	$92.08_{\pm 0.79}$	$659.01_{\pm 16.36}$	$19.53_{\pm 1.12}$
FixedTime	$297.13_{\pm 3.19}$	$49.26_{\pm 0.53}$	$457.21_{\pm 2.12}$	$86.27_{\pm 0.17}$	$682.55_{\pm 2.59}$	$17.76_{\pm 0.27}$
MaxPressure	$261.01_{\pm 4.20}$	$59.44_{\pm 1.51}$	$340.13_{\pm 1.69}$	$94.76_{\pm 0.19}$	$587.63_{\pm 30.92}$	$23.32_{\pm 2.69}$
SOTL	$416.88_{\pm 8.13}$	$8.46_{\pm 2.22}$	$424.67_{\pm 2.44}$	$90.02_{\pm 0.22}$	$624.19_{\pm 29.48}$	$21.81_{\pm 3.99}$
CoLight	-	-	$856.53_{\pm 451.32}$	$57.96_{\pm 30.08}$	-	-
FRAP	$258.21_{\pm 18.27}$	$\mathbf{63.49}_{\pm 8.19}$	$327.90_{\pm 23.28}$	$95.10_{\pm 1.17}$	$737.36_{\pm 84.12}$	$10.52_{\pm 5.98}$
MPLight	-	-	$\mathbf{297.00}_{\pm 8.32}$	$\mathbf{96.21}_{\pm 0.48}$	-	-
TLRP	$311.91_{\pm 35.18}$	$41.10_{\pm 12.51}$	$699.15_{\pm 224.30}$	$65.74_{\pm 17.31}$	$508.61_{\pm 88.51}$	$30.30_{\pm 7.00}$
TinyLight	$\mathbf{253.99}_{\pm 8.08}$	$62.16_{\pm 3.77}$	$310.62_{\pm 3.68}$	$95.79_{\pm 0.39}$	$\mathbf{489.93}_{\pm 19.76}$	$\mathbf{31.29}_{\pm 2.73}$

Table 1: The performance of all models on six real-world road networks (sec. = second, veh. = vehicle, min. = minute).

sub-graph extraction of TinyLight in Section 5.4. The experiments are conducted on CityFlow [Zhang *et al.*, 2019], an open-source simulator for realistic traffic environments.

5.1 Experimental Settings

Datasets. We evaluate our work on six road networks with real-world traffic flows, which are all publicly available.[4] Figure 3 presents a top view of these road networks, including three single intersections from Hangzhou, one 5×1 network from Atlanta, one 4×3 network from Jinan and one 4×1 network from Los Angeles. For datasets of Atlanta and Los Angeles, the intersection structures are heterogeneous. This prevents some baselines that require all intersections to be homogeneous. The traffic flows of Hangzhou and Jinan are obtained by surveillance cameras, and those of Atlanta and Los Angeles are collected from open vehicle trajectory data.[5] To measure the statistical variations of each model, we additionally create nine traffic flows for each road network, which are obtained by shifting the initial appearance time of all vehicles in the real-world records of traffic flow with a random noise $\Delta t \in [-60s, 60s]$.

Baselines. We adopt both rule-based ATSC solutions (EcoLight [Chauhan *et al.*, 2020], FixedTime [Miller, 1963], MaxPressure [Varaiya, 2013] and SOTL [Cools *et al.*, 2008]) and DRL-based ones (CoLight [Wei *et al.*, 2019b], FRAP [Zheng *et al.*, 2019] and MPLight [Chen *et al.*, 2020]) as our baselines. For the ablation study, we compare TinyLight with its variant that randomly selects paths to form the sub-graph, named TLRP (TinyLight with Random Path).

5.2 Traffic Efficiency

We measure the traffic efficiency of all models with two indexes: the travel time (per vehicle) and the throughput (per minute). The first index evaluates the average time required for a vehicle to complete its trip. The second index counts on average how many vehicles complete their trips in a minute. Both of these indexes are widely adopted in previous works on traffic signal control [Wei *et al.*, 2020].

The results of all models on these two indexes are presented in Table 1. It can be observed that TinyLight achieves advanced performances on both indexes. Specifically, our model requires the lowest travel time on five datasets and holds the highest throughput on four of them. Even for datasets where TinyLight has not reached the best score, its performance still remains competitive. For instance, although the average throughput of TinyLight on Atlanta dataset is not the best, it only falls behind FRAP by a margin of 1.33 vehicle per minute. On Jinan dataset, it is also the closest one to MPLight, the baseline which obtains the highest scores. Please note that comparing with all other DRL-based works, TinyLight is implemented with significantly fewer resource

[4] https://traffic-signal-control.github.io/

[5] https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

consumption (we will analyze this issue quantitatively in Section 5.3). Nevertheless, our model still reaches competitive performance on these road networks with real-world traffic demands. This supports the feasibility of TinyLight: to find a model that is both well-performed and tiny in size, letting it implementable on cheap devices with limited resources.

5.3 Resource Consumption

We analyze the resource consumption of each model with two criteria: storage and computation. They together determine the minimum configuration of the required device. We measure the model's storage by its parameter size and the model's computation by its amount of floating-point operations (FLOPs). These values are chosen because (i) they can be compared quantitatively, and (ii) they are independent of irrelevant conditions such as the clock rate or the optimization flags on compilers. These make them also frequently adopted in other resource-critic domains [Vaswani *et al.*, 2017].

Figure 4 depicts the parameter sizes and FLOPs of each model on an intersection from Jinan dataset.[6] The values are presented in a log-log plot and the normalized scores of travel time for each model are represented with bars on top of each model. Comparing with rule-based models, although TinyLight has slightly larger scales of storage and computation (but still in the capacity of our target MCU), its performance is on average better than them. This is because rule-based works highly rely on human experience and can be too rigid when dealing with dynamic traffic flows. Comparing with other DRL-based models, TinyLight requires significantly fewer resources. Its storage is diminished by our entropy-minimized objective function, which ablates redundant edges proactively and makes its parameter size 19.32 times smaller than CoLight. As FRAP and MPLight are built up with convolutional operators, their parameters only exist in kernels and therefore are also small in scale. However, such operation incurs intensive parallel operations, which severely impact the model's response time (>1.5s on our target MCU) if GPU is not available. Instead, the computation of TinyLight is bounded by our super-graph, which intentionally avoids these complex operators and thus is 90.43 times smaller than FRAP and MPLight. To make these values more intuitive, we plot in Figure 4 the borderlines of 32KB ROM and 0.1s response time on ATmega328P (an MCU with merely 2KB RAM, 32KB ROM and a clock rate of 8MHz), and TinyLight is the only DRL-based work which satisfies both conditions. This manifests the applicability of TinyLight on devices with extremely limited resources.

5.4 Ablation Study

We attribute the performance of TinyLight to the procedure of sub-graph extraction, which helps us find a qualified policy network from massive candidates automatically. To support this argument, we further compare TinyLight with TLRP, a variant of itself that replaces the found path with randomly selected ones. Since many of our candidate features are obtained from previous works and our super-graph only involves

[6]Appendix C provides the detailed computation procedure.

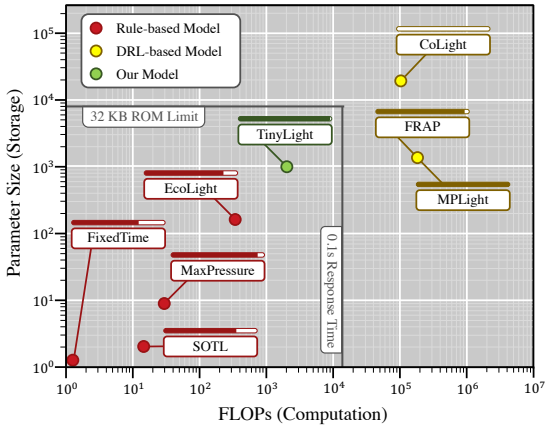


Figure 4: Log-log plot for each model's resource consumption. The bar on top of each model denotes its normalized score of travel time.

basic operations, TLRP can be regarded as a model that is empirically designed with the constraint of limited resources. It can be concluded from Table 1 that the variance of TLRP is much higher than that of TinyLight. For example, the variance of TLRP's travel time on Jinan dataset is 60.95 times higher than TinyLight. This reflects that manually designing a fixed network structure to fit all specific intersections is challenging. In contrast, TinyLight is able to adjust itself more smoothly to different intersections, as the model structure is automatically determined by the real traffic flows.

6 Conclusion

This paper presents TinyLight, the first DRL-based ATSC model that is designed for devices with extremely limited resources. We first construct a super-graph to cover a wide range of sub-graphs that are compatible with resource-limited devices. Then, we ablate edges in the super-graph automatically with an entropy-minimized objective function. This enables TinyLight to work on a standalone MCU with merely KB of memory and mediocre clock rate, which costs less than $5. We evaluate TinyLight on multiple road networks with real-world traffic demands and achieve competitive performance with significantly fewer resources. As traffic signal control is a field with broad application background, our work provides a feasible solution to apply ATSC on scenarios with limited budgets. Besides, our methodology can be trivially generalized to a wide range of fields that are cost sensitive.

We also acknowledge the limitations of our current solution and would like to point out some future directions for improvement. First, although TinyLight can be implemented on embedding devices, its training still requires general-purposed machines. We expect the training won't be frequent as traffic pattern often repeats periodically. However, a better idea is to directly enable on-chip model re-training. Second, currently we are focusing on small-scaled problems as our interested scenarios are those with limited budgets. However, larger problems are also common in the world. Dealing with them is more challenging as the searching space is much wider. Therefore, a more effective sub-graph extraction method is expected under this condition [Yang *et al.*, 2021].

Acknowledgments

This work was supported by Natural Science Foundation of China (No. 61925603), The Key Research and Development Program of Zhejiang Province in China (2020C03004) and Zhejiang Lab.

References

[Abdelgawad *et al.*, 2015] Hossam Abdelgawad, Kasra Rezaee, Samah El-Tantawy, Baher Abdulhai, and Tamer Abdulazim. Assessment of adaptive traffic signal control using hardware in the loop simulation. In *ITSC*, 2015.

[Branco *et al.*, 2019] Sérgio Branco, André G Ferreira, and Jorge Cabral. Machine learning in resource-scarce embedded systems, FPGAs, and end-devices: A survey. *Electronics*, 8(11), 2019.

[Chauhan *et al.*, 2020] Sachin Chauhan, Kashish Bansal, and Rijurekha Sen. EcoLight: Intersection control in developing regions under extreme budget and network constraints. In *NeurIPS*, 2020.

[Chen *et al.*, 2020] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *AAAI*, 2020.

[Cools *et al.*, 2008] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. Self-organizing traffic lights: A realistic simulation. In *Advances in Applied Self-organizing Systems*. 2008.

[Elefteriadou *et al.*, 2019] Lily Elefteriadou, Pruthvi Manjunatha, Sivaramakrishnan Srinivasan, Scott Washburn, Yafeng Yin, Xi Duan, Marian Ankomah, Yinan Zheng, Thomas Chase, Tyler Valila, et al. Before and after-implementation studies of advanced signal control technologies in Florida. Technical report, 2019.

[Elsken *et al.*, 2019] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 20(1), 2019.

[Hornik *et al.*, 1989] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 1989.

[Hunt *et al.*, 1981] PB Hunt, DI Robertson, RD Bretherton, and RI Winton. SCOOT - A traffic responsive method of coordinating signals. Technical report, 1981.

[Jiang *et al.*, 2021] Qize Jiang, Jingze Li, Weiwei Sun, and Baihua Zheng. Dynamic lane traffic signal control with group attention and multi-timescale reinforcement learning. In *IJCAI*, 2021.

[Liu *et al.*, 2020] Qianhui Liu, Haibo Ruan, Dong Xing, Huajin Tang, and Gang Pan. Effective AER object classification using segmented probability-maximization learning in spiking neural networks. In *AAAI*, 2020.

[Miller, 1963] Alan J Miller. Settings for fixed-cycle traffic signals. *Journal of the Operational Research Society*, 14(4), 1963.

[Oroojlooy *et al.*, 2020] Afshin Oroojlooy, MohammadReza Nazari, Davood Hajinezhad, and Jorge Silva. AttendLight: Universal attention-based reinforcement learning model for traffic signal control. In *NeurIPS*, 2020.

[Rizzo *et al.*, 2019] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. Time critic policy gradient methods for traffic signal control in complex and congested scenarios. In *SIGKDD*, 2019.

[Tang *et al.*, 2019] Keshuang Tang, Manfred Boltze, Hideki Nakamura, and Zong Tian. *Global Practices on Road Traffic Signal Control: Fixed-time Control at Isolated Intersections*. Elsevier, 2019.

[Varaiya, 2013] Pravin Varaiya. The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, volume 2 of *Complex Networks and Dynamic Systems*. Springer, 2013.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Wei *et al.*, 2019a] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash V. Gayah, Kai Xu, and Zhenhui Li. PressLight: Learning max pressure control to coordinate traffic signals in arterial network. In *SIGKDD*, 2019.

[Wei *et al.*, 2019b] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. CoLight: Learning network-level cooperation for traffic signal control. In *CIKM*, 2019.

[Wei *et al.*, 2020] Hua Wei, Guanjie Zheng, Vikash V. Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *SIGKDD Explor.*, 22(2), 2020.

[Xing *et al.*, 2021] Dong Xing, Qianhui Liu, Qian Zheng, and Gang Pan. Learning with generated teammates to achieve type-free ad-hoc teamwork. In *IJCAI*, 2021.

[Yang *et al.*, 2021] Long Yang, Qian Zheng, and Gang Pan. Sample complexity of policy gradient finding second-order stationary points. In *AAAI*, 2021.

[Yu *et al.*, 2020] Zhengxu Yu, Shuxian Liang, Long Wei, Zhongming Jin, Jianqiang Huang, Deng Cai, Xiaofei He, and Xian-Sheng Hua. MaCAR: Urban traffic light control via active multi-agent communication and action rectification. In *IJCAI*, 2020.

[Zhang *et al.*, 2019] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *WWW*, 2019.

[Zheng *et al.*, 2019] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. Learning phase competition for traffic signal control. In *CIKM*, 2019.