IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 26, 2019, accepted February 16, 2019, date of publication February 21, 2019, date of current version March 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900360

Dynamic Local Vehicular Flow Optimization Using Real-Time Traffic Conditions at Multiple Road Intersections

SOOKYOUNG LEE[1], MOHAMED YOUNIS[2], (Senior Member, IEEE), AISWARYA MURALI[1], AND MEEJEONG LEE[1]

[1]Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, South Korea

[2]Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County, Baltimore, MD 21250, USA

Corresponding author: Meejeong Lee (lmj@ewha.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) through the Ministry of Education under Grant 2016R1D1A1B03930635.

ABSTRACT Dynamic management of vehicular traffic congestion to maximize throughput in urban areas has been drawing increased attention in recent years. For that purpose, a number of adaptive control algorithms have been proposed for individual traffic lights based on the in-flow rate. However, little attention has been given to the *traffic throughput maximization problem considering real-time road conditions from multiple intersections*. In this paper, we formulate such a problem as maximum integer multi-commodity flow by considering incoming vehicles that have different outgoing directions. Then, we propose a novel adaptive traffic light signal control algorithm which opts to maximize traffic flow through and reduce the waiting time of vehicles at an intersection. The proposed algorithm adjusts traffic light signal phases and durations depending on real-time road condition of local and neighboring intersections. Via SUMO simulation, we demonstrate the effectiveness of the proposed algorithm in terms of traffic throughput and average travel time.

INDEX TERMS Adaptive traffic light control, k-commodity flow problem, traffic flow maximization, dynamic traffic management.

I. INTRODUCTION

Due to rapid urbanization, the density of vehicles on certain roads has grown at a high rate, causing increased traffic congestion, accidents, fuel consumption, CO_2 emission and travel delay [1]. These issues have motivated increased attention from the research community and led to the development of a variety of vehicular traffic management (VTM) strategies. One of the emerging strategies is to realize the notion of an intelligent transportation system (ITS) that relies on information collected using various sensing technologies such as safety CCTV, traffic video cameras, piezo-electric sensors, inductive loops, etc., to monitor road conditions and alert motorists through overhead message signs. However, such conventional surveillance methods used in ITS are not effective due to limited coverage and high maintenance cost. In order to overcome these shortcomings and collect real-time accurate traffic data, the ITS concept is augmented with wireless communication technologies leading to the development of vehicular ad-hoc network (VANET). Moreover, wireless sensor networks (WSNs) have been also exploited to improve traffic light timing [2]–[4].

Popular VTM strategies designed for improving traffic condition on urban roads can be categorized into two groups. The first group focuses on traffic data collection, e.g., using VANETs, and applying re-routing algorithms which detour vehicles based on the data [5]–[10]. The approaches vary based on when to respond to changes in the traffic pattern and how to determine the detour paths. Meanwhile, research in the second group explores adaptive traffic light control (ATLC) by taking advantage of advances in wireless communication technologies and computing resources to enable the consideration of real-time traffic conditions of each road [11]–[25]. The focus of most ATLC work has been mainly on increasing traffic throughput while considering a single intersection; only few studies consider

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

multiple intersections for traffic flow maximization [3], [4]. Unlike prior work, this paper tackles the *vehicular traffic flow maximization (TFM) problem while considering real-time condition at a target intersection and its neighboring intersections in the four directions, and models the TFM as a maximum integer multi-commodity flow problem (MCF).*

Since vehicles entering a road network of interest usually intend to go in various outgoing directions, *a flow of vehicles that have the same entry and exit points*, can be mapped into an individual commodity in a flow network. In addition, the number of vehicles belonging to the same flow, i.e., a group of vehicles that have the same departing and arriving positions, can be matched with a demand of a particular commodity. We map a road network into a flow network presented as a directed graph $G = (V, E)$, where V and E include all intersections and road segments, respectively. Based on G we show that the considered TFM problem can be mapped to MCF with conditions for *capacity*, *flow conservation*, and *demand satisfaction*. We then formulate a MCF model considering waiting vehicles for a green light heading towards a particular direction from an intersection i_C.

In order to solve the formed MCF problem, we introduce *an adaptive traffic light phase and duration optimization algorithm that factors in the road conditions of multiple intersections*(ATOM) as a means for reducing congestion and idle green durations. Basically, ATOM strives to increase traffic throughput and decrease waiting time at an intersection i_C depending on the real-time road conditions at i_C and its adjacent intersections. Furthermore, the traffic throughput increment and the waiting time decrement are dynamically adjusted and balanced by considering the level of road congestion around local and neighboring intersections. To collect traffic data, various communication technologies between on-board units (OBUs) on vehicles (V2V), between OBUs and road-side units (RSUs) at i_C (V2I) or between RSUs (I2I), are to be employed to track the traffic statistics at i_C, such as the number of incoming vehicles, the waiting time for all directions, etc.

Based on the level of congestion inferred from the collected traffic data, *ATOM applies distinct traffic light control approaches to strike a balance between increasing the number of passing vehicles, and reducing the average waiting time at i_C*. When roads around i_C experience low traffic volume, ATOM focuses on eliminating idle green time in order to reduce waiting time. Meanwhile, under heavy traffic conditions, the focus of ATOM is on relieving congested at nearby roads. In both cases, ATOM opts to adjust traffic light cycle and/or to optimize the selection and order of green light phases and their durations based on the monitored traffic data locally and at nearby intersections. The simulation results confirm the effectiveness of ATOM in adapting to traffic conditions and balancing system (throughput) and user (delay) interest.

The rest of the paper is organized as follows. Related work is covered in Section II. In Section III, the assumed system model is discussed and the problem is formally defined and formulated. The details of the proposed ATOM are provided in Section IV. The validation results are presented in Section V. The paper is finally concluded in Section VI.

II. RELATED WORK

As pointed out in the previous section, VTM systems found in the literature can be categorized into two groups, namely, dynamic vehicular rerouting and adaptive traffic light control. In this section, relevant work in these two groups is discussed. Since ATOM strives to optimally operate traffic light signals, the second group is covered in more details.

A. DYNAMIC VEHICLE REROUTING

To enable dynamic adjustment of routes Milojevic and Rakocevic [5] have proposed a distributed congestion detection algorithm based on V2V communication. The algorithm quantifies the traffic congestion level without any support of infrastructure and local authorities in providing real-time traffic data. A vehicle estimates congestion based on its speed and the duration for which such speed is sustained. Then, each vehicle V_i broadcasts its congestion estimate; upon hearing from its neighbors, V_i aggregates the neighbor estimates with its own to accurately assess the traffic conditions. The frequency of sharing traffic estimates is adaptively set according to the congestion level observed by the vehicle itself and obtained from other vehicles in the vicinity.

Meanwhile, some work has focused on re-routing strategies to relieve traffic jams and increase throughput [6], [7]. The objective in [6] is to reduce traffic density measured as the number of vehicles per minute over time. Congestion is mitigated by dynamic road pricing over the entire city in order to control the traffic flow. Road tolls are adjusted depending on the instantaneous change of traffic density tracked through inter-vehicle data sharing. The idea is to motivate drivers to use least-cost travel paths and thus avoid congested road segments. On the other hand, Pan *et al.* [7] have studied re-routing decisions to reduce the travel time. Multiple criteria for detour path selection are proposed, namely, random, entropy balance, or flow balance.

Moreover, autonomous route selection has researched in the realm of controlling autonomous vehicles [8]–[10]. Azimi *et al.* [8] have proposed a V2V intersection protocol to reduce the waiting time and increase throughput. They also extended their work in [9] to factor in GPS readings. Meanwhile, the focus of [10] is more on safety, collision avoidance and efficiency in terms of computation complexity while autonomous vehicles are passing at an intersection.

B. ADAPTIVE TRAFFIC LIGHT CONTROL

Work that focuses on ATLC can be classified into two categories based on whether a single or multiple intersections are being considered in determining signal timing. For a single intersection, the objective is to optimally schedule the green signal. For example in [11] the traffic light controller is assumed to know the destination of the vehicles coming at the intersection; two green light scheduling strategies are proposed, namely, least minimum and least average distance to destination with the objectives of reducing average waiting/travel time. Similarly, Pandit *et al.* [12] model the signal timing at one intersection as a job scheduling problem on processors, where jobs and processors correspond to platoons of vehicles and traffic light, respectively. Both approaches mainly focus on reducing travel delay based on real-time traffic. Meanwhile the focus of [13] is on reducing the total number of stops during the entire travel and thus ameliorate CO_2 emission. Unlike ATOM, these approaches do not strive to increase traffic throughput.

Moreover, Kwatirayo *et al.* [14] opt to improve the conventional pre-timed traffic light signals where the timing of G/Y/R/all-R cycle (GYRCT) is determined based on average traffic load. The approach enables the support of multiple traffic patterns, e.g., cycles within the day, where the duration of GYRCT is adjusted based on the anticipated traffic pattern. However, the approach is quasi-static in nature and does not adapt to unanticipated traffic fluctuations. Meanwhile, Hu and Wang [15] strive to decrease the average waiting time of vehicles crossing an intersection by considering upstream and downstream traffic density. A wireless sensor network is used to monitor the traffic condition. Like [15], Zhou *et al.* [2] rely on street-mounted sensor nodes to assess traffic and use the collected data to adjust all possible sequences of green lights in order to improve the waiting time, number of stops, and vehicle density. On the other hand, the focus of [16] is on reducing the time and space complexity for solving the traffic signal control problem while achieving the same goal of improving the average waiting time. Like ATOM, the approaches of [2], [15] try to optimize both green light sequence and length; however, they do not consider traffic data collected from multiple intersections.

In the second category of work, multiple neighboring intersections are considered in the traffic flow optimization. For example, rule-based reinforcement learning ATLC is presented in [17], where the traffic lights of neighboring intersections coordinate locally. The work is extended in [18] where an additional hierarchical observer/controller component is proposed at the regional level in order to better optimize the ATLC operation. Moreover, multi-agent based algorithms have been applied to traffic light systems [19]–[25]. Collotta *et al.* [19] employ multiple fuzzy logic controllers, interconnected using IEEE 802.15.4 technology and dynamically order phases and calculates green time while factoring turns. In addition, Elgarej *et al.* [20] employs sensors to monitor traffic volume variations, based on which they use a distributed multi-agent system to find the shortest green period during a vehicle trip so that the experienced waiting time at intersections is minimized. Multi-agent reinforcement learning algorithms are exploited in [21]–[25] where the reactions by local and nearby intersections are considered to adjust the traffic lights timing; however these approaches require higher computational complexities than ATOM and hence are less practical.

Moreover, Zhou *et al.* [2] have extended their approach in by considering traffic condition from neighbor intersections as well as local traffic volume when determining the duration of next green light in each intersection [3]. Similarly, Faye *et al.* [4] consider multiple intersections to optimize both green light sequence and length. Again, these multi-intersection approaches focus on the minimization of waiting time; unlike ATOM, vehicular throughput is not factored in and no flexible combinations of traffic light signals are considered. We compare the performance of ATOM to approach of [3], [4] and [19] in Section V.

Like ATOM, both SCATS and TRANSYT [36]–[39] opt to reduce the number of stops and consequently the trip delay, in city settings. ATOM and SCATS control traffic light signals (TLSs) in collaboratively at multiple intersections based on real-time traffic information; meanwhile TRANSYT precomputes phases and signal timings offline considering only intersections individually without coordination. Moreover, ATOM and SCATS both can control the cycle length and phase while TRANSYT adjusts phases given a set of cycle length. Like ATOM, SCATS is a microscopic system, which adjusts TLSs by analyzing individual vehicles' movement. However, TRANSYT is a macroscopic model which does not consider individual vehicles' movement. In addition, ATOM can be applied to a real road network in a distributed method as well as a centralized way unlike SCATS and TRANSYT.

III. PROBLEM STATEMENT AND FORMULATION

In this paper we strive to maximize traffic flow by controlling traffic lights at a target intersection i_C while considering the collected real-time vehicular traffic condition at its neighboring intersections. In addition, we consider a grid road network, as seen in Figure 1(a), where a vehicle at i_C moves towards one of the four adjacent intersections in the north, south, west and east direction of i_C, hereafter denoted as i_C^N, i_C^S, i_C^W, and i_C^E, respectively. However, we do not restrict the number of lanes and the length of a road segment between two adjacent intersections. Moreover, we do not consider pedestrians and crosswalks.

We represent the road network as a flow network via a directed graph $G = (V, E)$ where V includes the intersections, i.e., $V = \{i_C, i_C^N, i_C^S, i_C^W, i_C^E\}$ and E consists of road segments, i.e., directed edges between i_C and other intersections in V$\backslash i_C$. In other words, E is divided into E^{in} and E^{out}, each of which includes directed edges going to and coming out of i_C, namely, $E^{in} = \{e(i_C^N i_C), e\left(i_C^S, i_C\right), e\left(i_C^W, i_C\right), e(i_C^E i_C)\}$ and $E^{out} = \{e(i_C,i_C^N), e(i_C,i_C^S), e(i_C,i_C^W), e(i_C,i_C^E)\}$. Based on G we represent the monitored traffic at i_C as *demands* of a vertex $v \in V$ and *lower capacity bounds for* an edge $e \in E$, as seen in Figure 1(b) and (c).

By knowing the motion direction of vehicles at an intersection, one can estimate how many vehicles are coming into i_C from each of the four directions and vice versa. For instance, among 12 vehicles incoming into i_C^N seen in Figure 1(b), seven vehicles are supposed to drive towards i_C.

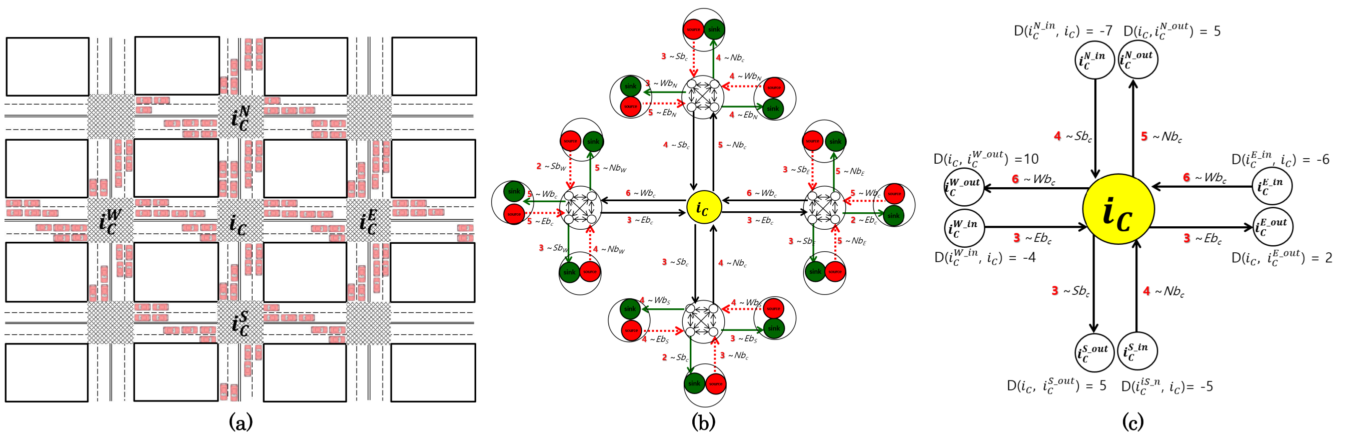

i_C^N
i_C^W
i_C
i_C^E
i_C^S
sink
source
$D(i_C^{N\_in}, i_C)$ = -7
$D(i_C, i_C^{N\_out})$ = 5
$D(i_C, i_C^{W\_out})$ =10
$D(i_C^{E\_in}, i_C)$ = -6
$D(i_C^{W\_in}, i_C)$ = -4
$D(i_C, i_C^{E\_out})$ = 2
$D(i_C, i_C^{S\_out})$ = 5
$D(i_C^{S\_in}, i_C)$= -5
4 ~Sb
5 ~Nb
6 ~Wb
3 ~Eb
3 ~Sb
4 ~Nb
(a)
(b)
(c)

FIGURE 1. TFM can be reduced to a *maximum integer multi-commodity flow problem* (MCF). The traffic flow around at an intersection $\mathbf{i_C}$ shown in (a) can be mapped into a flow network represented as a directed graph $G = (V, E)$, V and E is a set of intersections and road segments between intersections, respectively, as seen in (b). Part (c) additionally shows traffic volume which comes in or goes out of $v \in V$ and also the traffic volume waiting for a green light in each road segment $e \in E$.

Then we denote such incoming traffic into i_C as *negative demands*, i.e., $D(i_C^{N_{in}}, i_C) = -7$, as seen in Figure 1(c). On the other hand, outbound traffic for any of four directions leaving i_C can be denoted as *positive demands*. For example, in Figure 1(b), three, six and four vehicles are coming to i_C and five vehicles are routed to the north direction of i_C and thus $D(i_C, i_C^{N_{out}}) = 5$. In order to simplify the presentation, each neighbor intersection of i_C, i.e., $i_C^x, x \in \{\mathrm{N, S, W, E}\}$, is represented as two vertices as seen in Figure 1(c), one for traffic heading to i_C from x and one leaving i_C towards x, each of which is denoted as $i_C^{x_{in}}$ and $i_C^{x_{out}}$, respectively.

Based on the two types of *demands*, we define a local flow of vehicles at i_C that is hereafter denoted as F_i, by a pair of source src_i and destination dst_i and the number of vehicles in the flow n_i, where $0 \leq n_i \leq min\left(|D\left(src_i, i_C\right)|, |D(i_C, dst_i)|\right)$ Then there are 12 incoming flows F_i's at an intersection i_C that can be possible in Figure 1(a), hereafter denoted as $F_i = (src_i, dst_i, n_i), i = 1, \ldots, k$ (=12) where $src_i, dst_i \in \{i_C^N, i_C^S, i_C^W, i_C^E\}$ and $src_i \neq dst_i$. Then each F_i in a road network can be mapped into an individual commodity K_i in a flow network, $K_i = (s_i, t_i, d_i)$, where s_i and t_i are the source and sink of commodity i and d_i is the demand, and each of which is mapped into src_i, dst_i, and n_i, respectively. Based on G and F_i we formulate TFM as MCF by showing that for each edge $e_i \in E$, F_i satisfies the following three constraints that MCF holds [26]–[29]:

1) CAPACITY CONSTRAINTS

In the defined flow network G, the accumulated flow, i.e., the sum of F_i's on an edge $e \in E$ cannot be more than a *full capacity* or *upper bound of a capacity* of e denoted as $Ubs(e)$, which is usually a fixed value determined by the physical infrastructure, e.g., the number of lanes and the length of e. In addition, we consider real-time traffic data, namely, the number of delayed vehicles on e which have been waiting for a green signal in order to cross i_C, hereafter denoted as $NDV(e)$. $NDV(e)$ may vary depending on the condition of e, with a higher value of $NDV(e)$ indicating that a smaller number of additional vehicles can travel over e. For example, there are 3 east bound vehicles waiting on $e(i_C^W i_C)$ in Figure 1(a); therefore $NDV(e(i_C^W i_C)) = 3$ as noted in Figure 1(b). Also, in the opposite direction $NDV(e(i_C i_C^W))$ is 6 since six vehicles are currently staying on the link between i_C and i_C^W. Finally, an edge e can accommodate in-flow capacity based on the difference between $NDV(e)$ and $Ubs(e)$. In other words, the following equation holds in G.

$$0 \leq \sum\nolimits_{i=1}^{k} F_i(u, v), \quad \forall e(u, v) \in E \leq Ubs(e) - NDV(e)$$

2) FLOW CONSERVATION AT i_C

In addition, TFM opts to serve all vehicles in every flow F_i coming into i_C for the desired direction. Therefore, the sum of F_i's entering i_C must equal that of F_i's exiting i_C; in other words, TFM fulfills the *flow conservation* condition of MCF.

$$\sum\nolimits_{i=1}^{k} F_i(u, v) = \sum\nolimits_{i=1}^{k} F_i(vu)$$

where u $\in \left\{i_C^N, i_C^S, i_C^W, i_C^E\right\}$ and $v = i_C$

3) DEMAND SATISFACTION

Finally, TFM is to satisfy demands in a *multi-commodity flow problem*; that is the sum of incoming F_i's to G via src_i must be equal to that of outgoing F_i's from G via dst_i, i.e., for each i the following equation holds:

$$F_i(src_i, i_C) = F_i(i_C dst_i) = n_i,$$

where $0 \leq n_i \leq min\left(|D\left(src_i, i_C\right)|, |D(i_C, dst_i)|\right)$

Therefore, it can be concluded that TFM is indeed MCF. Thus, the optimization objective can be formulated as:

$$Maximize \sum\nolimits_{i=1}^{k} F_i(src_i, i_C)$$

$$subject\ to\ (1)\ 0 \leq \sum\nolimits_{i=1}^{k} F_i(u, v), \leq Ubs(e) - NDV(e),$$

$$\sum_{i=1}^{k} F_i(u, i_C) = \sum_{i=1}^{k} F_i(i_C u),$$
$$u \in \left\{ i_C^N, i_C^S, i_C^W, i_C^E \right\} \tag{1}$$

and

$$F_i(src_i, i_C) = F_i(i_C dst_i) = n_i, 0 \le n_i$$
$$\le min\left((src_i, i_C)\,|, |D(i_C, dst_i)|\right)$$

ATOM strives to maximize the total vehicle traffic volume flowing into i_C by adjusting traffic light phase and duration. Since MCF is a known NP-complete problem for integer flows, ATOM pursues heuristics. The proposed algorithm will be detailed in the next section. Since we do not consider pedestrians, the procedure of the optimization does not count the minimum green time for pedestrians to cross roads at i_C. Moreover, ATOM considers driver-based vehicles as well as self-driving cars; for the former, it is assumed that the driving direction of an individual vehicle is provided by a driver at each intersection, e.g., using V2I technologies or by an app on the driver's cell phone.

IV. THE ATOM APPROACH

The key feature of ATOM is to employ two distinct traffic flow optimization approaches depending on the dynamically changed vehicle density on the road. ATOM consists of two traffic light agents, namely, the traffic flow watcher and evaluator (TFW), and the traffic light signal adjuster (TLSA). The agents operate in parallel and simultaneously cooperate with each other as illustrated in Figure 2. In this section, the architecture of ATOM is first explained and the notations and preliminary formulations are introduced. Then the details of ATOM are explained in the balance of the section.

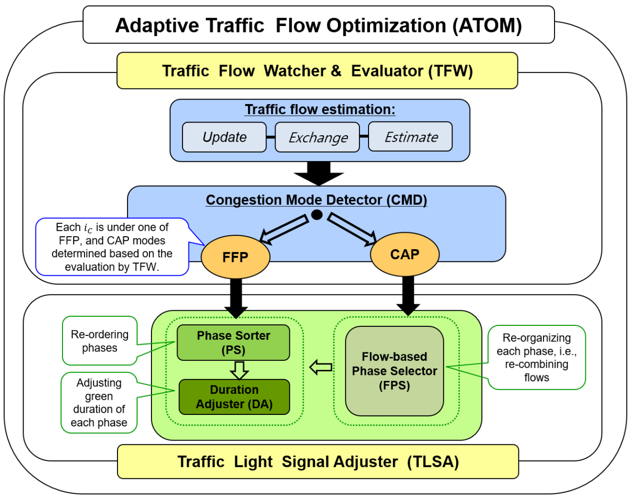

Adaptive Traffic Flow Optimization (ATOM)
Traffic Flow Watcher & Evaluator (TFW)
Traffic flow estimation:
Update
Exchange
Estimate
Congestion Mode Detector (CMD)
Each i_c is under one of FFP, and CAP modes determined based on the evaluation by TFW.
FFP
CAP
Re-ordering phases
Phase Sorter (PS)
Adjusting green duration of each phase
Duration Adjuster (DA)
Flow-based Phase Selector (FPS)
Re-organizing each phase, i.e., re-combining flows
Traffic Light Signal Adjuster (TLSA)

FIGURE 2. The architecture of ATOM highlighting the interaction among two agents, i.e., traffic flow watcher and evaluator (TFW), and traffic light signal adjuster (TLSA).

A. ARCHITECTURE OF ATOM

As stated above, ATOM opts to improve the overall traffic throughput and to diminish the average vehicle waiting at i_C. ATOM employs two agents, namely TFW and TLSA. TFW monitors dynamic traffic changes locally and at nearby intersections and estimates the volume of incoming traffic heading towards i_C, and the conditions on the outbound roads from i_C. The data collection can be performed in practice using various V2V, V2I, and I2I technologies such as road sensors, 802.11p, 802.16, i.e., WiMAX or cellular networks like LTE-V, or 3GPP Cellular-V2X(C-V2X) [30]–[33]. Then, TFW assesses the level of congestion on a monitored road. Basically, the *congestion mode detector* (CMD) of TFW categorizes the condition of i_C into two modes, namely, *free flow period* (FFP), and *continuously adjusted period* (CAP). In addition, TFW periodically evaluates the impact of the traffic optimization done by TLSA and feeds back the results to TLSA to improve the performance. Meanwhile, TLSA optimizes traffic flow by applying different strategies depending on the congestion level at i_C, assessed by TFW, as explained below:

- *Free Flow Period (FFP):* This represents the time period in which the overall volume of vehicles that stay or pass through i_C is low and thus the traffic throughput at i_C is mainly affected by the length of a cycle during which every flow takes its turn for green light at least once. In FFP, a short cycle is generally preferred for decreasing the average waiting time for vehicles passing i_C in the various directions since the total number of vehicles served by i_C is small. Therefore, TLSA during FFP, denoted hereafter as *ffp*-TLSA, computes an appropriate length for the traffic light cycle at i_C in which all F_i's are served by a green light once, and adjusts the green signal order and duration to favor the directions with relatively more in-flows. Furthermore, FFP is divided as three sub-levels depending on which a cycle length, phases and green light durations are adjusted in an incremental way.
- *Continuously Adjusted Period (CAP):* When TFW observes that the vehicle count on the monitored roads continuously grows and approaches a level where TLSA does not show further improvement for traffic throughput at i_C, CMD switches the mode at i_C into CAP. In order to relieve road congestion and increase throughput, TLSA under CAP, henceforth denoted as *cap*-TLSA first analyzes traffic volume increment per in-flow on forwarding roads and foresees future traffic status per flow on driving roads at i_C in collaboration with i_C's adjacent intersections. Then *cap*-TLSA strives to find an optimized combination of out-flows that are associated with green signal phases; *cap*-TLSA then optimizes the order and duration of these phases in order to maximize the average traffic throughput. Fundamentally, *cap*-TLSA identifies and eliminates unnecessary green duration computed for free flows during subsequent cycles and instead gives priority for green time to more congested flows heading to freer outbound road. Starvation for green time in the free flows is prevented using threshold.

TFW and TLSA can be executed at a traffic light controller or on a regional traffic control center. With the former option, TFW and TLSA can be implemented in a popular hardware kit like Raspberry Pi or Arduino. Then the agents at the individual intersections collaborate with each other in a decentralized manner to coordinate the operation of the traffic signals. In addition, a centralized realization of ATOM could be cloud-based or done by a server. In this paper we consider both implementation scenarios.

B. NOTATIONS AND FORMULATIONS

The notations used in the paper falls in four categories; the first includes *road configuration parameters at an intersection* i_C, the second covers a set of *vehicular traffic flow related parameters*, and the variables used for *traffic light signal management,* finally the congestion level evaluation parameters are grouped in the third and fourth categories respectively. For quick reference, a summary of the notations is provided in Appendix A.

1) ROAD CONFIGURATION

A road segment S that ends at an intersection i_C can be one of two types; an inbound from i_C^x towards i_C and an outbound from i_C to i_C^x, hereafter denoted as $S(i_C^x i_C)$ and $S\left(i_C, i_C^x\right), x \in \{N, S, W, E\}$, respectively. The following attributes describes S:

- $Len(S)$: is the length of S which influences the number of vehicles that can be simultaneously located on S.
- $CP(S)$: denotes the full capacity of S *in terms of how many vehicles can physically be on* S, i.e., $\frac{Len(S)\times number\ of\ lanes\ on\ S}{avg.vehicle\ length}$.

$Len(S)$ and $CP(S)$ are all constant and cannot be changed by ATOM.

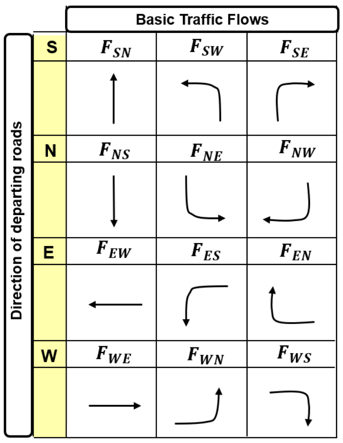


FIGURE 3. 12 types of flows are handled at i_C, each of which is composed of {departing, turning, and arriving at i_C}.

2) VEHICULAR TRAFFIC FLOW

Based on the three basic signals *go-left* (*GL*), *go-straight* (*GS*) and *go-right*(*GR*), 12 local flows passing at i_C are represented as seen in Figure 3. We denote $F_i = (src_i, dst_i, n_i), i = 1, \ldots, k$ (=12) where $src_i, dst_i \in \{i_C^N, i_C^S, i_C^W, i_C^E\}$ as $F(i_C^b, i_C^a)$, where $b, a \in \{N, S, W, E\}$ or F_{ba} in short. For example, the flow of vehicles which depart from i_C^S and go straight towards i_C^N is denoted as $F(i_C^S i_C^N)$ or F_{SN}. In addition, the flows can be categorized depending on the used roads before and after i_C; for instance F_{SN}, F_{SW}, and F_{SE} are included in a flow of vehicles departing from i_C^S whose throughput would affect congestion on the road segment $S(i_C^S i_C)$. Similarly, F_{SW}, F_{NW}, and F_{EW} are flows of vehicles coming into a road segment $S(i_C i_C^W)$. A set of in-flows towards i_C is hereafter denoted as S_{inflow}, where $|S_{inflow}|$, denoted as N_{in_flow}, equals 12 as enumerated in Figure 3.

The parameters associated with each $F_{ba} \in S_{in\_flow}$ are listed below. These parameters are updated by TFW at the end of a cycle CYC_t.

- $ph_t(F_{ba})$: represents a phase ph for which a flow F_{ba} is served during CYC_t. TLSA may adjust flow membership in a phase according to the changes in the traffic volume.
- $dur_t(F_{ba})$: denotes the green signal duration for F_{ba}, i.e., the time period during which vehicles in F_{ba} are allowed to pass through i_C in CYC_t. This value constitutes the green period of the phase which F_{ba} belongs to, i.e., $ph_t(F_{ba})$.
- $TH_t(F_{ba})$: is the number of vehicles of a flow F_{ba} which have departed from i_C during $dur_t(F_{ba})$. The sum of $TH_t(F_{ba})$ for all flows at i_C becomes a throughput of i_C during CYC_t.
- $NDV_t(F_{ba})$: represents the number of vehicles that got delayed at i_C, i.e., could not pass during CYC_{t-1} and have to be considered in CYC_t.
- $Delay_t(F_{ba})$: reflects the average waiting time for the vehicles in F_{ba} waiting for a green light in CYC_t. i.e., it represents the average waiting time experienced by vehicles $\in NDV_t(F_{ba})$.
- $Speed_t(F_{ba})$: represents the average driving speed of vehicles $\in F_{ba}$. The lower the value is, the higher the probability of congestion on the road becomes. $Speed_t(F_{ba})$ can be measured by TFW based on the speed information sent by vehicles, e.g. using V2I.
- $NAV_t(F_{ba})$: is the number of additional vehicles to an incoming flow F_{ba} towards i_C from its neighboring intersection i_C^b between the green signals in the previous cycle CYC_{t-1} and in the current cycle CYC_t. Actually, $\sum_{\forall b}(NDV_t\,(F_{ba}) + NAV_t(F_{ba})) = n_i$ for $F_i = (src_i, dst_i, n_i)$ in CYC_t.
- $NEV_t(F_{ba})$: is the expected number of vehicles joining F_{ba}, i.e., the flow which will head to i_C^a from i_C^b through i_C in the next cycle CYC_{t+1}. It is measured by using the flows towards i_C in CYC_t on the road $S(i_C^b i_C)$. and pre-known turning information of vehicles at i_C.

The values of $TH_t(F_{ba})$, $NDV_t(F_{ba})$, $Delay_t(F_{ba})$, $Speed_t(F_{ba})$, and $NAV_t(F_{ba})$ can be obtained by using various V2I technologies between OBU and the signal controller. In addition, $NEV_{t+1}(F_{ba})$ is obtained using I2I communication between two neighboring intersections.

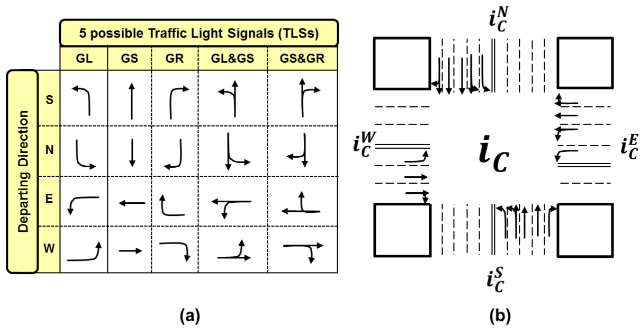


FIGURE 4. **(a) represents a set All_{TLS} of 20 possible TLSs for each lane and (b) shows an example intersection, where 15 different TLSs $\in All_{\text{TLS}}$ are used for the various lanes.**

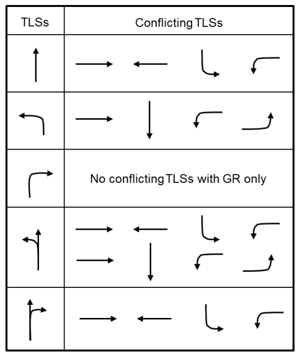


FIGURE 5. **Regardless of direction, conflicting TLSs should be avoided during the formation of a phase.**

3) TRAFFIC LIGHT SIGNAL MANAGEMENT

The parameters in this category are mainly used by TLSA and represent the attributes employed while controlling a traffic light signal (TLS) at an intersection i_C. We consider five types of TLSs, namely *GL*-only, *go-left-and-go-straight* (*GL&GS*), *GS-only*, *go-straight-and-go-right* (*GS&GR*) and *GR*-only. Therefore, ATOM monitors traffic status, and determines congestion level and green duration based on 12 flows while phase formation is performed based on 20 TLSs seen in Figure 4(a). Hence, ATOM becomes more flexible and applicable to various types of intersections in terms of the number of attached roads and TLSs, for instance ATOM can be also applied to 3-leg intersections.

- CYC_t: denotes a traffic signal cycle in which every F_{ba} is served at least once with a green signal. Basically, the same F_{ba} may experience green signals more than once in a certain CYC_t according to the results of applying TLSA.
- $TLSpL(i_C^x)$: lists a TLS per lane on the inbound road $S(i_C^x i_C)$ among the five distinct TLSs $\{GL$ only, $GL\&GS$, GS only, $GS\&GR$, GR only$\}$. Figure 4(a) shows a set All_{TLS} of 20 possible TLSs considering directions and Figure 4(b) shows an example intersection, where 15 distinct TLSs in All_{TLS} are used on the road $S(i_C^x i_C)$, $x \in \{N, S, W, E\}$.
- ph: refers to a traffic light phase, i.e., a combination of two or more TLSs $\in TLSpL(i_C^x), \forall x$ in non-conflicting directions and whose green lights can be allowed at the same time at an intersection i_C. Thus, a set All_{ph} of all possible phases ph's is determined depending on $TLSpL(i_C^x), \forall x$ defined at i_C and non-conflicting TLSs seen in Figure 5. Since every F_{ba} is at least once served with green in CYC_t, $\bigcup_{\forall ph \in CYC_t} ph \supset \bigcup_{\forall b \neq a \in \{N,S,E,W\}} F_{ba}$.
- $dur_t(ph)$: reflects a green period allowed for a phase ph in a cycle CYC_t. Such a period has an impact on the number of passing vehicles $\in ph$ at i_C in CYC_t and the operation of TLSA. For example, under the FFP mode, $dur_t(ph)$ is assigned based on the relative in-flow traffic volume at i_C in CYC_t.
- $dur(CYC_t)$: denotes the length of CYC_t which equals a sum of $dur_t(ph)$ for all phases in a cycle, i.e., $\sum_{\forall ph \in CYC_t} dur_t(ph)$. Assuming that every flow F_{ba} is served only once in each CYC_t, $dur(CYC_t)$ may imply the longest time for a group of vehicles to wait for a green light since it has arrived at i_C as seen in Figure 6(a). In addition, $dur(CYC_t)$ may influence how quickly TLSA can respond to relieve traffic congestion; therefore, TLSA adjusts the value of $dur(CYC_t)$ to optimize the traffic throughput. For example, in the FFP mode *ffp*-TLSA strives to optimize value of $dur(CYC_{t+1})$ at i_C as well as the order of ph's in CYC_t based on the real-time road condition in CYC_t. Meanwhile, *cap*-TLSA cares about each value of $dur_t(ph)$ which finally affects $dur(CYC_t)$; recall that $dur(CYC_t) = \sum_{\forall ph \in CYC_t} dur_t(ph)$.

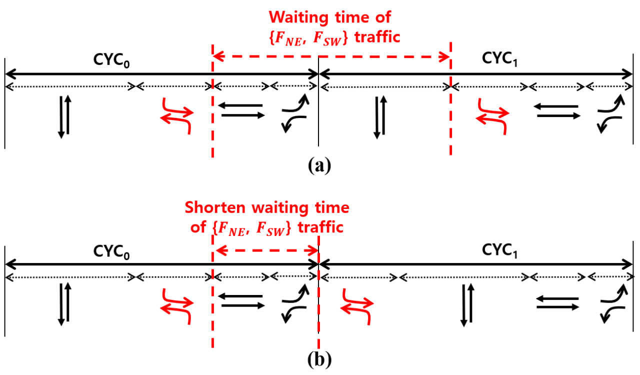

Waiting time of $\{F_{NE}, F_{SW}\}$ traffic
CYC_0
CYC_1
(a)
Shorten waiting time of $\{F_{NE}, F_{SW}\}$ traffic
CYC_0
CYC_1
(b)

FIGURE 6. **An example of CYC_t which consists of four phases shows the impact of an order of the phases,ph in each cycle.**

- $Speed_t(ph)$: represents the average speed of driving vehicles that belong to a phase ph in CYC_t. It is equal to $\frac{\sum_{\forall F_{ba} \in ph} Speed_t(F_{ba})}{|\forall F_{ba} \in ph|}$.
- $PH(CYC_t)$: lists the phases in CYC_t which may vary depending on *ffp*-TLSA or *cap*-TLSA. The entire set All_{ph} of all possible phases is considered by TLSA to form $PH(CYC_t) \subset All_{ph}$. Since the order of phases in CYC_t has an influence on increasing or decreasing

the traffic throughputs of particular flows, TLSA strives to find the best order of phases in a cycle as well as $dur_t(ph)$ or $dur(CYC_t)$. As seen in Figure 6(b), although each phase in CYC_1 has the same duration as the previous cycle CYC_0, placing a phase of $\{F_{NE}, F_{SW}\}$ ahead in CYC_1, will reduce a waiting time for vehicles in the two flows, F_{NE} and F_{SW} which may be more congested than others.

4) CONGESTION LEVEL EVALUATION

The parameters in this category reflect metrics measured by TFW and used by TLSA at the beginning of each cycle to assess traffic conditions and to evaluate the effect of the employed optimization measures.

- $FOR_t(F_{ba})$: represents the *forecasted road occupancy ratio by vehicles in*F_{ba}*relative to its capacity,* i.e.,

$$FOR_t\,(F_{ba}) = Delay_t\,(F_{ba}) \times \frac{(NDV_t\,(F_{ba}) + NAV_t\,(F_{ba}))}{CP(S(i_C^b, i_C))}$$

  where $S(i_C^b, i_C)$ is the road segment that F_{ba} occupies passing i_C. Since occupancy ratio will be used to assess how crowded a road S is and how long congestion may last, the waiting time of vehicles $\in F_{ba}$ is factored in.
- $CL_t(F_{ba})$: represents *a congestion level or traffic demand for* F_{ba} *in* CYC_t *relative to other flows* passing at i_C. It factors in the forecasted occupancy ratio in CYC_t and the expected number of incoming vehicles in F_{ba} in CYC_{t+1} as well as the throughput of F_{ba} in CYC_{t-1}, i.e., $FOR_t(F_{ba}), NEV_t\,(F_{ba})$ and $TH_{t-1}\,(F_{ba})$, respectively, $CL_t(F_{ba})$ is calculated as follows:

$$CL_t\,(F_{ba}) = \frac{FOR_t\,(F_{ba})}{\sum\limits_{\forall F \in i_C} FOR_t\,(F)} + \frac{TH_{t-1}\,(i_C)}{TH_{t-1}\,(F_{ba})} + \frac{NEV_t\,(F_{ba})}{NEV_t\,(i_C)}$$

  In other words, $CL_t\,(F_{ba})$ is proportional to $FOR_t(F_{ba})$, and $NEV_t\,(F_{ba})$ and inversely proportional to $TH_{t-1}\,(F_{ba})$; thus, the linear combination will enable a high relative throughput to make up for a high ratio of vehicle pile-up, captured by a high occupancy. Since $Delay_t(F_{ba})$ and $TH_t(F_{ba})$ are determined by TLSA, a value of $CL_t(F_{ba})$ reflects a real-time change of traffic demand in F_{ba}. In addition, considering delay and throughput together reflects scenarios that include various combination of delay and throughput performance.
- $CL_t(ph)$: denotes the congestion or traffic demand level of a phase ph, which is determined by $CL_t\,(F_{ba})\,, \forall F_{ba} \in ph$, i.e., $CL_{ph} = \sum_{\forall F_{ba} \in ph} CL_t(F_{ba})$. Basically, $CL_t(ph)$ measures an aggregated congestion level of vehicular flows in a phase ph. Thus a higher value implies more congested phases or roads.

Algorithm 1 in Appendix B summarizes how TFW maintains the congestion evaluation parameters.

C. ADAPTIVE TRAFFIC FLOW OPTIMIZATION ALGORITHM

The objectives of ATOM are to maximize the vehicular traffic throughput and to minimize the average travel time. Such optimization can be expressed as maximizing $\sum_{\forall F} TH_t(F_{ba})$ and also minimizing $\sum_{\forall F} Delay_t(F_{ba})$ and $\sum_{\forall F} NDV_t(F_{ba})$, for $\forall t$ at i_C. In order to achieve these objectives, ATOM strives to identify the congestion level of i_C depending on which it adaptively controls traffic light signals at i_C by ordering phases and computing optimized green timing. Such signal timing and ordering optimization is based on real-time local and neighboring intersections' road status. The overall procedure is split into two parts, namely, TFW and TLSA; each part operates independently by autonomous agents. In this section, each agent is separately explained and their interactions are also presented.

1) TRAFFIC FLOW WATCHER AND EVALUATOR (TFW)

TFW is composed of two modules: (i) traffic flow estimation, and (ii) congestion mode decision, as explained below.

a: TRAFFIC FLOW ESTIMATION

The main goal of this module is to estimate changes for each flow F_{ba} passing a target intersection i_C in terms of its volume and delay. The estimation is based on collection and analysis of traffic flow variations around multiple neighboring intersections of i_C, i.e., i_C^N, i_C^S, i_C^W, i_C^E. For carrying out this task, TFW performs three operations, namely, "*update*", "*exchange*", and "*estimate*". First, TFW tries to keep the latest status for the twelve $F'_{ba}s \in S_{in\_flow}$ seen in Figure 3. Basically, TFW *updates* flow-related parameters, $NDV_t(F_{ba})$, $Delay_t(F_{ba})$, $TH_t(F_{ba})$, $Speed_t\,(F_{ba})$, $NAV_t\,(F_{ba})$ and $NEV_t(F_{ba})$ each time a cycle CYC_t ends. The update can be done in practice by deploying on-road sensors or traffic cameras, or by exploiting V2I communication between vehicles and traffic light controllers on the local inbound roads, $S\left(i_C^b, i_C\right), b \in \{\mathrm{N, S, E, W}\}$. In addition, TFW *reports/receives* the updated time-varying status of flow to/from the neighboring intersections, $i_C^x, x \in \{\mathrm{N, S, W, E}\}$. The traffic data transfer between intersections can be done via RSU-to-RSU, or I2I wireless communication. After hearing from neighboring intersections, TFW becomes informed of the traffic condition on the roads leading to i_C and estimates the incoming flows during CYC_{t+1}, i.e., $NEV_t\,(F_{ba})$.

To elaborate, TFW first updates traffic flow status on the road, for instance $S(i_C^E i_C)$ seen as a dark shaded area in Figure 7 using local resources at i_C. Then using the traffic data obtained from i_C^E, TFW tries to estimate the expected volume of incoming vehicles from i_C^E, in the next cycle, CYC_{t+1}, which corresponds to a sum of three flows F_{NW}, F_{EW}, and F_{SW} at i_C^E, hereafter denoted as $\{F_{NW}, F_{EW}, F_{SW} |\, i_C^E\}$-seen as a double line in Figure 7. Then, the anticipated number of vehicles which will join F_{EW}, F_{EN}, and F_{ES} in CYC_{t+1}, is estimated with the help of known driving direction information from the individual vehicles at i_C, as stated in Section III. Therefore, for each direction $a \in \{\mathrm{N, S, W}\}$, TFW computes

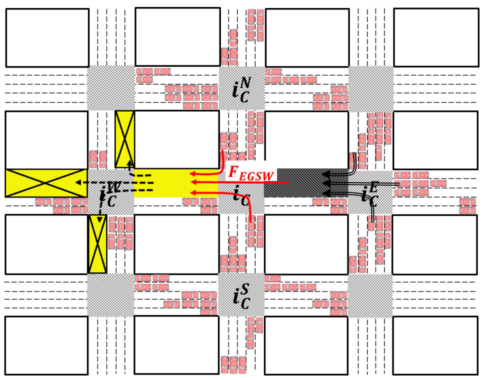


FIGURE 7. TFW monitors *before* and *after* roads condition passing $\mathbf{i_C}$ in collaboration with neighboring intersections, $\mathbf{i_C^x}$, $\mathbf{x \in \{N, S, W, E\}}$.

$NEV_t(F_{Ea})$, which is equal to $\sum_{b \neq a \in \{N,S,E,W\}} TH_t(F_{ba})$. The updated value of $NEV_t(F_{Ea})$ will be used by *cap*-TLSA to optimize the performance as explained later in this section.

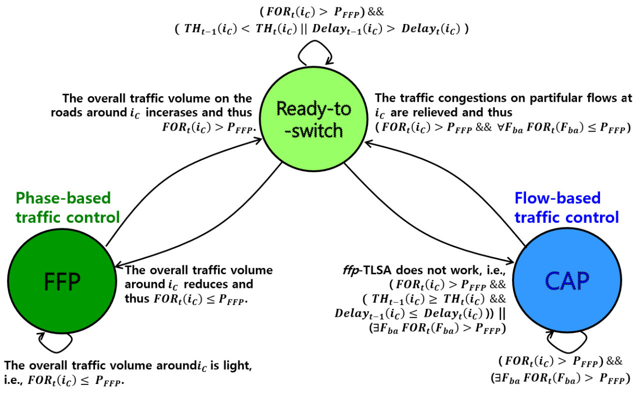
Moreover, in order to maximize the number of vehicles crossing i_C, TFW also estimates the conditions (availability) of the outbound roads from i_C i.e., $S\left(i_C, i_C^a\right), a \in \{N, S, W, E\}$ considering the changes of $Delay_t(F_{ba})$, $NDV_t(F_{ba})$, $TH_t(F_{ba})$ and $NEV_t(F_{ba})$ in two subsequent cycles and shares the estimates with TLSA. For instance, the traffic throughput of F_{EW} in the next cycle on the road $S\left(i_C^E, i_C\right)$ will be affected by how crowded $S\left(i_C, i_C^W\right)$ is, seen as a bright area in Figure 7, as well as the count of the coming vehicles on $S\left(i_C^E, i_C\right)$, i.e., $NEV_t(F_{EW})$. Therefore, TFW tries to assess the ability of $S\left(i_C, i_C^W\right)$ for handling the outgoing west-bound flows based on the observed status of $S\left(i_C, i_C^W\right)$ by i_C^W. TFW makes such assessment using the congestion level of the road $S\left(i_C, i_C^W\right)$, i.e., $CL_t\left(S\left(i_C, i_C^W\right)\right) = \sum_{\forall F_{Ea} \in F_{out}} CL_t(F_{Ea})$ where $F_{out} = \{F_{EN}, F_{EW}, F_{ES} | i_C^W\}$ are provided by i_C^W. The value of $CL_t\left(S\left(i_C, i_C^W\right)\right)$ hints the congestion level on the three road segments, $S\left(i_C^W, (i_C^W)^N\right)$, $S\left(i_C^W, (i_C^W)^W\right)$, and $S\left(i_C^W, (i_C^W)^S\right)$ seen as X-marked rectangles in Figure 7. Therefore, the traffic optimization at i_C performed by TLSA is based on the real-time traffic status on its neighboring roads as well as its local ones.

b: CONGESTION MODE DECISION (CMD)

In ATOM, congestion is assessed based on *how many vehicles are waiting* to cross an intersection, *how long they are waiting*, and *how many of them actually cross.* For determining the congestion level in CYC_t at i_C, CMD considers three factors, namely, occupancy ratio, delay, and number of vehicles that crossed i_C, i.e., throughput with respect to i_C, hereafter denoted as $FOR_t(i_C)$, $Delay_t(i_C)$, and $TH_t(i_C)$, respectively, and computed as the average value over all in-flows for i_C. Thus

$$FOR_t(i_C) = \left(\sum\nolimits_{\forall F_{ba} \in i_C} FOR_t(F_{ba})\right)/N_{in\_flow},$$
$$Delay_t(i_C) = \left(\sum\nolimits_{\forall F_{ba} \in i_C} Delay_t(F_{ba})\right)/N_{in\_flow}$$
$$TH_t(i_C) = \left(\sum\nolimits_{\forall F_{ba} \in i_C} TH_t(F_{ba})\right)/N_{in\_flow}.$$

Using the three parameters, CMD determines the mode for i_C, denoted as $Mode_t(i_C)$, to be either *FFP* or *CAP*. In order to assure that i_C is controlled under the right mode and avoid the unnecessary switching between the modes, we exploit a *window-based threshold* using a tentative mode called *ready-to-switch* as seen in Figure 8. Initially when the overall traffic volume on the roads around i_C is light, $Mode_t(i_C)$ is set to FFP. When the overall road traffic is continuously growing and becomes larger than a pre-defined threshold (P_{FFP}), i_C switches to a *ready-to-switch* state before turning into CAP when the increase persists, as seen in Figure 8. During the *ready-to-switch* state, the traffic light is still controlled by the same control strategy that is *ffp*-TLSA, however CMD begins to additionally monitor *changes of traffic status on individual flows* F_{ba}.



$(FOR_t(i_C) > P_{FFP})$ && $(TH_{t-1}(i_C) < TH_t(i_C) \,||\, Delay_{t-1}(i_C) > Delay_t(i_C))$
Ready-to -switch
The overall traffic volume on the roads around i_C incerases and thus $FOR_t(i_C) > P_{FFP}$.
The traffic congestions on partifular flows at i_C are relieved and thus $(FOR_t(i_C) > P_{FFP}$ && $\forall F_{ba}\, FOR_t(F_{ba}) \leq P_{FFP})$
Phase-based traffic control
Flow-based traffic control
FFP
CAP
The overall traffic volume around i_C reduces and thus $FOR_t(i_C) \leq P_{FFP}$.
ffp-TLSA does not work, i.e., $(FOR_t(i_C) > P_{FFP}$ && $(TH_{t-1}(i_C) \geq TH_t(i_C)$ && $Delay_{t-1}(i_C) \leq Delay_t(i_C)))$ || $(\exists F_{ba}\, FOR_t(F_{ba}) > P_{FFP})$
The overall traffic volume aroundi_C is light, i.e., $FOR_t(i_C) \leq P_{FFP}$.
$(FOR_t(i_C) > P_{FFP})$ && $(\exists F_{ba}\, FOR_t(F_{ba}) > P_{FFP})$

FIGURE 8. State diagram shows transitions between *FFP*, and *CAP*. A *Ready-to-switch* state avoids unnecessary mode switching.

Under the *ready-to-switch* state, if $TH_t(i_C)$ or $Delay_t(i_C)$ is further improved by *ffp*-TLSA then $Mode_t(i_C)$ is kept unchanged. Otherwise, if *ffp*-TLSA does not relieve congestion at i_C in terms of $TH_t(i_C)$ and $Delay_t(i_C)$ then $Mode_t(i_C)$ is switched to CAP. During CAP, traffic lights are controlled by *cap*-TLSA which mainly employs flow-based traffic optimization and the CAP status is maintained while $FOR_t(i_C) > P_{FFP}$ && $\exists F_{ba}$, where $FOR_t(F_{ba}) > P_{FFP}$. $Mode_t(i_C)$ may be switched back to a *ready-to-switch* state when the occupancy for all in-flows, i.e., $FOR_t(F_{ba}), \forall F_{ba}$ are under P_{FFP}. Figure 8 shows the state transition of $Mode_t(i_C)$ and Algorithm 2 in Appendix B outlines the CMD procedure.

2) TRAFFIC LIGHT SIGNAL ADJUSTER (TLSA)

Based on the identified mode, TLSA applies different traffic light control approaches depending on FFP or CAP. Under FFP, the TLSA module mainly focuses on decreasing the vehicle waiting time at i_C by removing unnecessary (idle) green duration. Meanwhile, in the CAP mode, TLSA opts to maximize throughput by reducing road congestion at i_C. Thus the optimization strategy varies between *ffp*-TLSA and *cap*-TLSA. For both modes, a *phase sorter* (PS), and a *green duration adjuster* (DA) are applied, while *cap*-TLSA additionally employs a *flow-based phase selector* (FPS), as seen in Figure 2.

ffp-TLSA: When $Mode_t\,(i_C)$ is deemed to be FFP, TLSA exploits three strategies in an incremental way; ① adjusting the cycle length, i.e., $dur(CYC_t)$, ② reorganizing the set of phases, $PH(CYC_t)$, and ő adjusting the green duration of each phase, i.e., $dur_t(ph)$, $\forall ph \in CYC_t$. If the traffic flows freely on the roads around i_C, for example the passing traffic volume is significantly low relatively to the road capacity, then *ffp*-TLSA focuses on reducing the waiting time and simply sets the recommended value of $dur(CYC_{t+1})$ for the next cycle based on the average delay and number of phases in the current cycle using the following equation:

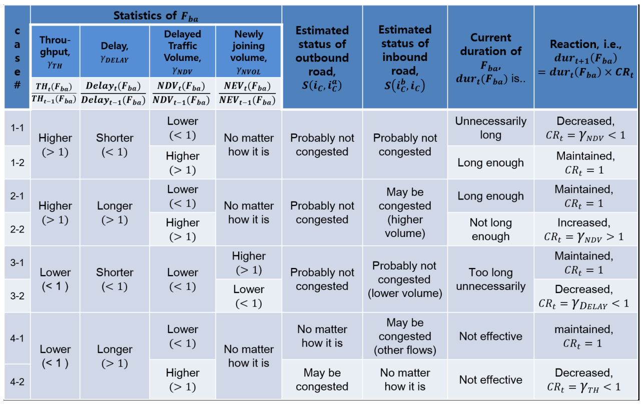
$$dur\,(CYC_{t+1}) = Delay_t\,(i_C) \times |PH(CYC_t)|$$

where $Delay_t\,(i_C)$ denotes the average delay at i_C in CYC_t.

When reduction in waiting time is actually not achieved and also the incoming vehicle count into i_C gets larger, *ffp*-TLSA additionally exploits re-ordering the phases in $PH(CYC_t)$ based on a congestion level in each phase, i.e., $CL_t\,(ph)$. When the second approach cannot improve the average waiting time at i_C, i.e., $Delay_t\,(i_C)$, *ffp*-TLSA furthermore recompute duration of individual phase for the next cycle, i.e., $dur_{t+1}\,(ph)$ as $\sum_{\forall F_{ba} \in ph}((NDV_t\,(F_{ba}) + NAV_t\,(F_{ba}) + NEV_t\,(F_{ba})) \times \frac{Len(S)}{Speed_t(F_{ba})})$, which is sufficient for all vehicles to cross i_C in the next cycle. Then, $dur\,(CYC_{t+1})$ is automatically changed as $\sum_{\forall ph \in PH(CYC_t)} dur_{t+1}\,(ph)$. It is important to note that the incremental application of the above three strategies, adjusting $dur(CYC_t)$, reorganizing $PH(CYC_t)$, and computing $dur_t(ph)$ in order is to limit the scope of the change in the traffic signal operation, simplify the implementation, and sustain stability of the traffic signal controller.

cap-TLSA: Unlike the phase-based FFP mode, under *CAP* TLSA strives to control the vehicular volume per flow to reduce road congestion and thus increase traffic throughput. However, like *ffp*-TLSA, *cap*-TLSA factors in the anticipated volume of vehicles coming from neighboring intersections towards i_C in CYC_{t+1}. Basically, *cap*-TLSA considers three flow-based parameters, namely, $Delay_t(F_{ba})$, $NDV_t(F_{ba})$ and $TH_t(F_{ba})$ which capture the change trend for traffic flow during two consecutive cycles CYC_{t-1} and CYC_t at i_C. In addition, *cap*-TLSA exploits the foreseen congestion level of outbound road $\left(i_C, i_C^a\right)$, i.e., in a flow $F_{ba}, b, a \in \{N, S, W, E\}$, in CYC_{t+1}. Basically, extending the green signal duration for the flow F_{ba} will not improve the overall traffic throughput at i_C if the outbound road segment is crowded.

As seen in lines 8-30 of Algorithm 4 in Appendix B, *cap*-TLSA first estimates $dur_{t+1}(F_{ba})$ while considering changes in $Delay_t(F_{ba})$, $NDV_t(F_{ba})$, $TH_t(F_{ba})$ and $NEV_t(F_{ba})$ relative to those of the previous cycle, CYC_{t-1}. The estimation can be categorized into four cases, as seen in Figure 9. According to the identified case, *cap*-TLSA adjusts $dur_{t+1}(F_{ba})$ based on $dur_t(F_{ba})$ and optimizes its



case #	Statistics of F_{ba}				Estimated status of outbound road, $S(i_C, i_C^a)$	Estimated status of inbound road, $S(i_C^b, i_C)$	Current duration of F_{ba}, $dur_t(F_{ba})$ is..	Reaction, i.e., $dur_{t+1}(F_{ba}) = dur_t(F_{ba}) \times CR_t$
	Throughput, γ_{TH}	Delay, γ_{DELAY}	Delayed Traffic Volume, γ_{NDV}	Newly joining volume, γ_{NVOL}				
	$\frac{TH_t(F_{ba})}{TH_{t-1}(F_{ba})}$	$\frac{Delay_t(F_{ba})}{Delay_{t-1}(F_{ba})}$	$\frac{NDV_t(F_{ba})}{NDV_{t-1}(F_{ba})}$	$\frac{NEV_t(F_{ba})}{NEV_{t-1}(F_{ba})}$				
1-1	Higher (> 1)	Shorter (< 1)	Lower (< 1)	No matter how it is	Probably not congested	Probably not congested	Unnecessarily long	Decreased, $CR_t = \gamma_{NDV} < 1$
1-2			Higher (> 1)				Long enough	Maintained, $CR_t = 1$
2-1	Higher (> 1)	Longer (> 1)	Lower (< 1)	No matter how it is	Probably not congested	May be congested (higher volume)	Long enough	Maintained, $CR_t = 1$
2-2			Higher (> 1)				Not long enough	Increased, $CR_t = \gamma_{NDV} > 1$
3-1	Lower (< 1)	Shorter (< 1)	Lower (< 1)	Higher (> 1)	Probably not congested	Probably not congested (lower volume)	Too long unnecessarily	Maintained, $CR_t = 1$
3-2				Lower (< 1)				Decreased, $CR_t = \gamma_{DELAY} < 1$
4-1	Lower (< 1)	Longer (> 1)	Lower (< 1)	No matter how it is	No matter how it is	May be congested (other flows)	Not effective	maintained, $CR_t = 1$
4-2			Higher (> 1)		May be congested	No matter how it is	Not effective	Decreased, $CR_t = \gamma_{TH} < 1$

FIGURE 9. ***cap*-TLSA opts to adjust $\mathbf{dur_{t+1}(F_{ba})}$ based on $\mathbf{dur_t(F_{ba})}$ considering changes of traffic statistics during previous two cycles.**

phases, $PH\,(CYC_{t+1})$ by reorganizing them and adjusting length. The following discusses each of the four cases in Figure 9:

Case #1: implies relatively low road congestion even with increasing traffic volume. Thus, in this case, traffic throughput of F_{ba} at i_C increases in CYC_t in comparison to CYC_{t-1}, the associated $Delay_t(F_{ba})$ decreases, and more vehicles wait at i_C, i.e., $NDV_t(F_{ba})$ becomes larger. In other words, in this case the inbound and outbound roads are not congested and the volume of the incoming traffic is still within the road capacity. Therefore, *cap*-TLSA deems the current length of the cycle to be long enough. As a result, it maintains or reduces $dur_{t+1}\,(F_{ba})$ relative to the ratio of the remaining traffic volume between two subsequent cycles, i.e., $\frac{NDV_t(F_{ba})}{NDV_{t-1}(F_{ba})}$, denoted as γ_{NDV}. Hence, only if the waiting traffic volume is reduced, i.e., $\gamma_{NDV} < 1$, $dur_{t+1}\,(F_{ba})$ reduced as much as γ_{NDV}, otherwise the duration stays the same.

Case #2: corresponds to when both $TH_t(F_{ba})$ and $Delay_t(F_{ba})$ increase due to growth in traffic volume on road $S\left(i_C^b, i_C\right)$; yet no congestion on a forwarding road $S\left(i_C, i_C^a\right)$ is of concern given the high throughput for F_{ba}. In this case, *cap*-TLSA concludes that the extended delay may be caused by lower-than-expected driving speed on the road $S\left(i_C^b, i_C\right)$, not by the duration of the green signal. Thus, *cap*-TLSA increases or maintains $dur_{t+1}\,(F_{ba})$ depending on $NDV_t(F_{ba})$ like the case #1. In other words, only when the waiting traffic volume grows, i.e., $\gamma_{NDV} > 1$, *cap*-TLSA increase $dur_{t+1}\,(F_{ba})$ as $dur_t\,(F_{ba}) \times \gamma_{NDV}$ since no potential congestion is expected on an outbound road.

Case #3: are when the traffic demands on an inbound road diminishes. Then, *cap*-TLSA assumes that $dur_t\,(F_{ba})$ may be unnecessarily long because of the shorter $Delay_t(F_{ba})$, and consequently reduces $dur_{t+1}\,(F_{ba})$ only if the number of joining vehicles in F_{ba} during CYC_{t+1}, i.e., $NEV_{t+1}(F_{ba})$ is not expected to grow in comparison to CYC_t. The decrement is determined by $\frac{Delay_t(F_{ba})}{Delay_{t-1}(F_{ba})}$, denoted as γ_{DELAY}, and thus $dur_{t+1}\,(F_{ba}) = dur_t\,(F_{ba}) \times \gamma_{DELAY}$, where $\gamma_{DELAY} < 1$. Otherwise, i.e., more traffic is expected to come and join in F_{ba}, $dur_{t+1}\,(F_{ba})$ remains as $dur_t\,(F_{ba})$.

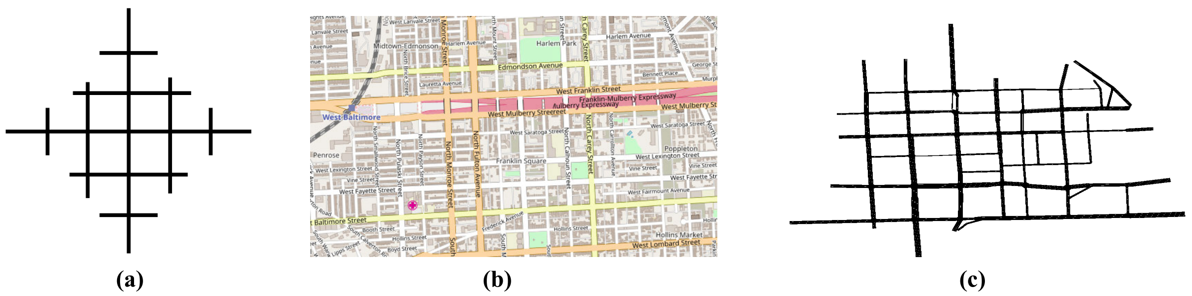


FIGURE 10. **The configurations considered in the simulation: (a) a road grid, and (b) and (c) represent a real road map imported from Baltimore area (1083×933 m^2). Detail info is presented in Table 1.**

Case #4: represents the situation when the throughput of in-flows F_{ba}, i.e., $TH_t(F_{ba})$ has been degraded in two subsequent cycles due to congestion on either road segment, inbound $S\left(i_C^b, i_C\right)$ or outbound $S\left(i_C, i_C^a\right)$. To cope with Case #4, *cap*-TLSA tries to figure out where the traffic jam is developing out of two possibilities. First, when observing reduced number of delayed (unpassed) vehicles after CYC_t compared to CYC_{t-1}, i.e., $\gamma_{NDV} < 1$, the drop in throughput can be attributed to congestion on the outbound road $S\left(i_C, i_C^a\right)$. Therefore, a change in the green signal duration is unwarranted and *cap*-TLSA keeps $dur_{t+1}\left(F_{ba}\right)$ similar to $dur_t\left(F_{ba}\right)$. On the other hand, the second possibility is when $NDV_t\left(F_{ba}\right)$ exceeds $NDV_{t-1}(F_{ba})$, i.e., $\gamma_{NDV} > 1$ which indicates that the inbound road segment $S\left(i_C^b, i_C\right)$ is getting congested as well. Therefore, *cap*-TLSA reduces the green time, i.e., makes $dur_{t+1}\left(F_{ba}\right)$ smaller than $dur_t\left(F_{ba}\right)$, since it is not being fully utilized. The reduction is set proportional to the observed drop in throughput, i.e., $\frac{TH_t(F_{ba})}{TH_{t-1}(F_{ba})}$, denoted as γ_{TH}. Thus,

$$dur_{t+1}\left(F_{ba}\right) = dur_t\left(F_{ba}\right) \times \gamma_{TH},$$

where $\gamma_{TH} < 1 \&\& \gamma_{NDV} > 1$

Upon computation of $dur_{t+1}\left(F_{ba}\right)$ for all flows, *cap*-TLSA opts to identify inefficient green light durations in terms of traffic throughput and exclude them from the next cycle. In other words, the F_{ba} whose computed $dur_{t+1}\left(F_{ba}\right)$ is less than the pre-determined minimum duration threshold gets excluded in the next cycle. The value is adjusted by considering road network configuration. Then *cap*-TLSA tries to reform $PH(CYC_t)$ by selecting flows which can be grouped in a phase, $ph \in ALL_{ph}$ such that the gap between $dur_{t+1}\left(F_{ba}\right), \forall F_{ba} \in ph$ is the least. The phase organization is carried out in the decreasing order of the congestion level of flows, i.e., $CL_t\left(F_{ba}\right)$ Therefore, more congested flows will thus be placed before less congested ones in the next cycle. The phase reorganization terminates when every flow is included in a phase. Algorithm 3 and 4 in Appendix B describe how *cap*-TLSA works for duration adjustment and phase selection, respectively.

V. PERFORMANCE EVALUATION

ATOM is validated through simulation. This section discusses the simulation environment, baseline approaches, performance metrics and results.

A. SIMULATION ENVIRONMENT AND PERFORMANCE METRICS

The simulation has been implemented in Python using SUMO [34]. Both a configurable road grid and a real road network are used in the experiments. The former enables assessing the effectiveness of ATOM under varying road length and width. As seen in Figure 10(a), the considered grid consists of 13 intersections, 44 road segments, with a variant of incoming and outgoing lanes. All road segments have the same length. Varying the segment length would change the vehicular capacity and consequently the performance of ATOM. In addition, a real road network of a mid-size city is considered. The map from Baltimore area is imported using open street maps as shown in Figure 10(b)-(c); such a network consists of 311 road segments and 137 junctions with traffic light signals. The relevant parameters are enumerated in Table 1. In other words, the performance of ATOM is validated using both synthetic and real road networks. For road network of the city of Baltimore, instead using sample traffic data set, the performance under various traffic conditions is studied; the rationale is that using sample traffic data would not be sufficient since it will not be covering all possible scenarios.

TABLE 1. **Road network information.**

Parameter	Grid network		Real road network
Total number of edges	88		311
Total number of lanes	176	440	2172
Edge length (m)	200	800	Vary in [1,40]
Total road length (m)	17600	70400	10470
Number of traffic light signals	13		137

Based on the configuration, the traffic condition can be varied by adjusting an *arrival rate* of vehicles inserted into the network per simulation time. The vehicle arrival follows an exponential distribution with average inter-arrival time (IAR) μ. In other words, IAR denotes a time interval between which two subsequent vehicles are injected to a road network. A lower μ, means higher arrival rate and consequently more vehicles on the road network and higher road occupancy. In addition, for each vehicle a pair of entry and exit points is randomly selected among the given roads in the grid or real road network. We study the performance based on two route selection methodologies, namely, the shortest travel distance and the Logit-model [35]. The travel path for a vehicle is calculated by SUMO DUAROUTER. The performance of ATOM is evaluated using the following four metrics at a target intersection i_C:

- *Service rate*: is a ratio of the number of the vehicles inside the road network out of the total vehicles trying to use the network. It represents the main performance of the traffic light control algorithms. The more efficient ATLC algorithm serves more vehicles at intersections and yields a higher service rate.
- *Average waiting time of vehicles* (seconds): is a measure of the time that vehicles wait on the average at i_C. The shorter the waiting time, the better the algorithm performs. It is equal to the average sum of all vehicles passing an intersection i_C.
- *Average driving speed* (km/h): equals the average of the driving speed for vehicles which finish their trip through the network. The speed of each vehicle is calculated by dividing its trip length by the time taken during journey. This metric gauges the end-to-end vehicular trip efficiency.
- *Average number of stopped vehicles*: denotes the average count of vehicles that had to wait for a green light at i_C. This metric is indicative for the traffic flow under different congestion levels.
- *Total number of vehicles travelling in the network*: represents how many vehicles are being travelling and thus reside in the network as simulation time runs. Given IAR, the road network is occupied by less vehicles with a more efficient ATLC algorithm applied.
- *Total number of vehicles completing their travel*: reports the number of vehicles that exited from the network. Ideally, all algorithms should be converged to the same value determined by IAR and simulation time, however, convergence time varies depending on the effectiveness of the algorithm.

B. BASELINE APPROACHES

The performance of ATOM is compared to that of three baseline approaches, namely, the WSN-based ATLC (WSN-ATLC) [3], the dynamic traffic light management based on Wireless Sensor Networks and multiple fuzzy logic controllers (FUZZY) [19], and the distribuTed and AdaPtive IntersectiOns Control Algorithm (TAPIOCA) [4]. All the baselines have the same objectives as ATOM which is to increase traffic throughput and decrease waiting time at intersections by adaptively traffic light controlling at the intersections. However, the factors which they exploit to achieve the goals are distinct and finally ends with different results. Detail of each algorithm is explained in Appendix C and Table 2 provides a comparative summary of ATOM to the three baselines based on the different factors used for traffic light control and their influence. As seen in Table 2, all baselines control multiple intersections except FUZZY which considers only single intersection i_C. In addition, all baselines have a smaller set of phases than ATOM, where TAPIOCA and WSN-ATLC do not consider a GR traffic light and FUZZY considers only the fixed set of four phases. Moreover, unlike WSN-ATLC and FUZZY, TAPIOCA and ATOM consider the outbound road conditions. TAPIOCA considers traffic volume difference between outbound and inbound roads only during phase selection while ATOM estimates outbound road status considering changes of passing/waiting/incoming traffic volume, and delay in two subsequent cycles during green duration computation as well as phase selection. This aspect of ATOM has significant influence on service rate since the unnecessarily used green lights can be avoided.

Another factor which sets ATOM apart is flow-based traffic management. Unlike the baselines, ATOM monitors, controls and evaluates vehicular traffic per flow in a CAP mode and thus during *cap*-TLSA it assigns vehicles in more congested flow with higher priority for a green time than those in free flow. Moreover, prioritizing vehicles per flow based on the number of vehicles that are waiting, arriving, departing and will be coming enables ATOM to be more effective than WSN-ATLC and TAPIOCA, which consider only waiting, arriving and departing vehicles, and FUZZY, which only considers the number of waiting vehicles.

C. SIMULATION RESULTS

We have simulated multiple levels of congestion by varying the vehicle arrival rates, where μ is picked in the range [0.2, 3] and IAR has been varied depending on a different road network configuration to simulate less or more congested networks. For example, in a grid network with 2 lanes per road segment, IAR is set to 0.5 second, and thus a total of 7200 vehicles are loaded into the configured road network during 3600 second simulation time. The cycle initially consists of 12 phases and its length is set to 120 seconds; consequently, each phase has a green light for 10 seconds. The simulation results are collected from the summary file created by SUMO. The results of the individual experiments are averaged over 10 runs and all results thus stay within 10% of the sample mean due to 90% confidence interval analysis.

1) GRID NETWORK WITH 2 LANES PER ROAD

Figure 11 represents the performance of algorithms in a 2-lane grid network. IAR varies between 0.5 and 1 with an increment of 0.1. Figure 11(a) shows that ATOM outperforms all baselines in terms of the service rate regardless of congestion levels in the road network. ATOM achieves 45% service rate with IAR = 0.5, which constitutes the most congested roads. Meanwhile TAPIOCA shows about 31% service rate at the same IAR. The performance gap between ATOM and TAPIOCA becomes larger as IAR grows and the traffic becomes lighter. This is because unlike WSN-ATLC, FUZZY and TAPIOCA, ATOM optimizes the combination and order of phases considering all possible TLSs and changes of traffic status on the roads in each cycle. ATOM also determines

TABLE 2. **Summary of ATOM, WSN-ATLC, FUZZY and TAPIOCA.**

Factor	ATOM	WSN-ATLC	FUZZY	TAPIOCA
Number of considered intersections	Multiple	Multiple	Single	Multiple
Adjustment of phase and duration	Both	Both	Both	Both
TLS control approach	Decentralized	Decentralized	Decentralized	Decentralized
Different strategies applied depending on a level of road congestion	Two level congestion controller such as FFP and CFP	Not considered	Not considered	Not considered
TLS patterns applicable	{GL only, GL&GS, GS only, GS&GR, GR only}	{GL only, GL&GS, GS only}	{GL only, GS&GR}	{GL only, GL&GS, GS only}
Outbound lane traffic condition	Considered by difference of passing traffic volume, delay, waiting volume between inbound and outbound roads during phase selection and duration computation both	Not available	Not available	Considered by only the difference of traffic volume between inbound and outbound roads only during phase selection
Depth of traffic monitoring - **Scope of monitored vehicles** - **Unit of monitored vehicles**	- Waiting, arriving, departing and will-be-arriving vehicles - Per phase or per flow - For phase selection and duration computation -	- Waiting, arriving, and departing vehicles - Per phase only 11 - For phase selection and duration computation	- Waiting vehicles only - Per phase only - For phase selection and duration computation	- Waiting, arriving, and departing vehicles - Per phase only - Only for phase selection

green duration while factoring in the outbound road condition by monitoring traffic status changes of the flows heading towards the road with respect to traffic throughput, delay and the expected number of incoming vehicles. It is noted that TAPIOCA performs worse than WSN-ATLC and FUZZY even though it estimates and reflects the outbound road condition. Such performance is because TAPIOCA does not consider that the delayed or arriving traffic volume may have changed when calculating green duration, and thus the computed time for a next green light may be unnecessarily long or insufficient.

Moreover, Figure 11(b) and (c) show the performance comparison of ATOM to baselines in terms of the average waiting time and the average number of stopped vehicles for a green light at intersections. As expected all approaches shorten the waiting time and reduces the number of vehicle stops when a road network is less congested with larger IAR. ATOM yields the least waiting time, mainly a result of the congestion-aware phase optimization. ATOM considers all possible combinations of movements for making a phase and forms an optimized phase that combines non-conflicting flows. It is noted that the effectiveness of WSN-ATLC is improved as the roads become less congested which is expected since it explicitly considers road availability in scheduling phases and computing green durations.

Figure 11(d) depicts the comparisons of the average driving speed of travelling vehicles under different levels of congestion. The results of each algorithm have been normalized to those of TAPIOCA. When traffic light signals in the network is managed by ATOM, vehicles can drive about 300% faster than by TAPIOCA with IAR $= 1$ and the average speed drops as the roads get more congested with IAR $= 0.5$. Overall ATOM outperforms all baseline approaches and enables the travelers to drive closer to the road's speed limit. Such distinct performance is because ATOM tries to relieve congestion ahead based on analysis of traffic status changes in the two previous cycles and thus provides vehicles with more space to keep their normal speed. Unlike the other approaches, TAPIOCA does not show much changes regardless of IARs. Such performance is attributed to TAPIOCA's green time computation method which only takes into account the maximum number of vehicles and ignores other factors like the number of vehicles that are expected to come in the next interval.

2) GRID NETWORK WITH 5 LANES PER ROAD

We have extended the grid configuration by increasing the number of lanes and length of each road segment to five lanes and 800m, respectively. Figure 12 shows the results. Given the increased road width and length in this setup,

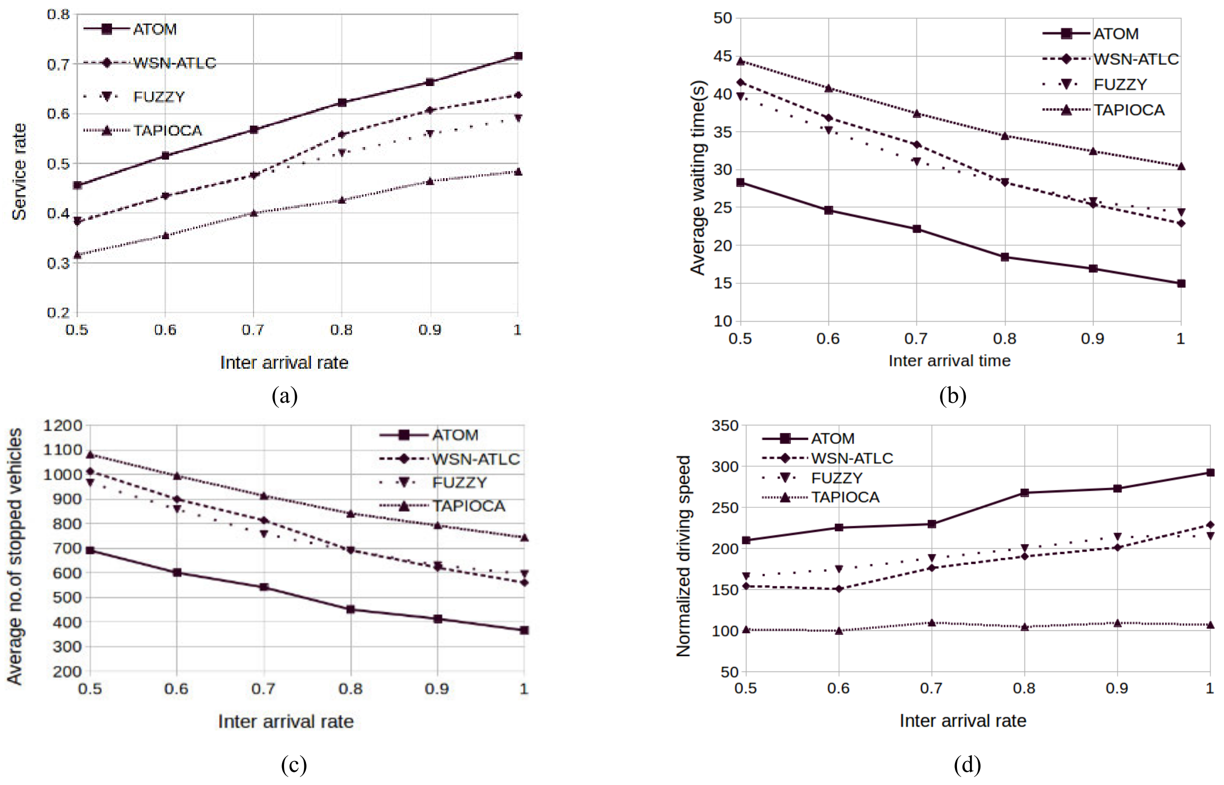


FIGURE 11. **Performance comparison of ATOM to WSN-ATLC, FUZZY and TAPIOCA in a grid network with 2 lanes on each road in terms of (a) service rate, (b) average waiting time, (c) average number of stopped vehicles while waiting for green light to pass i_C and (d) their normalized driving speed.**

the IAR values are picked between 0.6 and 1.4 in order to simulate levels of road congestion that resemble the 2-lane grid network. Comparing Figures 12(a) and 11(a) at IAR = 0.6, all approaches except WSN-ATLC show higher service rates in the 5-lane grid network than the 2-lane case. This is very much expected given the increased capacity. Meanwhile, the effectiveness of WSN-ATLC diminishes due to its limitation of selecting two lanes for compsoing a phase which finally degrades its service rate in the enlarged road network. In addition, in the 5-lane grid network the service rate of ATOM reaches up to 95% at IAR = 1.4; in fact it yields 75% at IAR = 0.6 which is higher than those of baselines at IAR = 1.4 as seen in Figure 12(a). Again, this is because of ATOM's better congestion assessment and consideration in determining the phase and green duration. Overall, ATOM makes tangible performance improvement for such a large road setup as IAR grows from 0.6 to 1.4, in comparison to the little improvement achieved by the baselines. The results in Figure 12(a) confirm the scalability of ATOM for large road networks.

Figure 12(b) and (c) show the average waiting time and the average number of stopped vehicles at intersections. Similar to the 2-lane grid network, ATOM significantly outperforms the alternatives, dropping the waiting time and number of stopped vehicles by up to 65%. ATOM achieves such distinct performance since it analyzes all movement combinations when forming a phase and allocates green duration to congested flows; this helps in reducing the delay and consequently diminishes the average number of vehicles in the queue waiting for green to move. As observed in the figures, ATOM shows about at least 60% better performance than FUZZY, TAPIOCA and WSN-ATLC when IAR is 0.6. ATOM also shows relatively the highest average driving speed of vehicles, which exceed by far all baselines, as seen in Figure 12(d). Overall ATOM yields shorter queues at intersections and enables higher motion speeds.

3) REAL ROAD NETWORK

Figure 13 shows the performance of ATOM and two baselines, WSN-ATLC and TAPIOCA in the real road network described in Figure 10(b)-(c). FUZZY is not compared to ATOM in this setup since Fuzzy assumes 4-leg intersections only and each intersection has a fixed set of phases which is {GL only, GS&GR} among 20 possibilities seen in Figure 4(a). Thus, it is hardly applicable to a real road network where various types of intersections and road configurations, such as 3-leg intersections, and one-way roads, to which the fixed four phases cannot be applied. As explained in Table 1, the considered Baltimore city based road network includes 137 intersections with traffic lights, and 311 roads each of which has a distinct setup in terms of the number of lanes and a road length that varies in [1] and [40]. Thus, a total number of lanes and a total

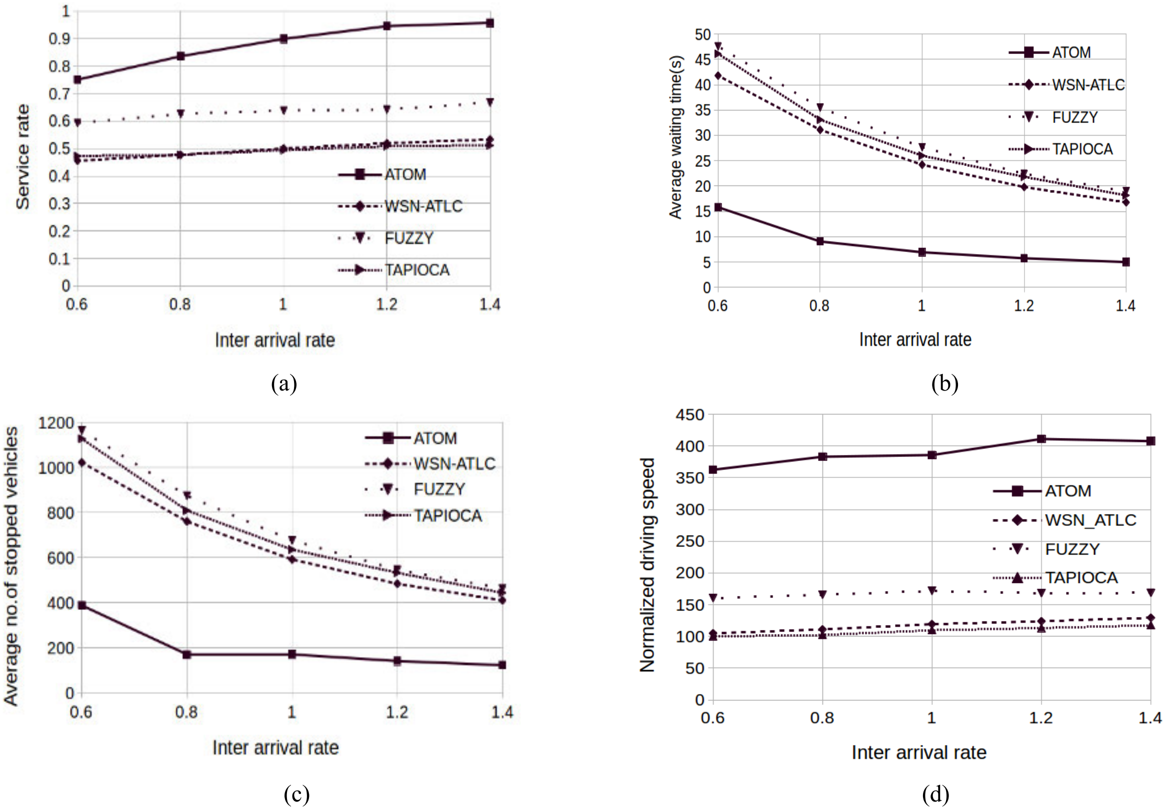


FIGURE 12. Performance comparison of ATOM to WSN-ATLC, FUZZY and TAPIOCA in a grid network with 5 lanes per road in terms of (a) service rate, (b) average waiting time, (c) average number of stopped vehicles and (d) normalized driving speed.

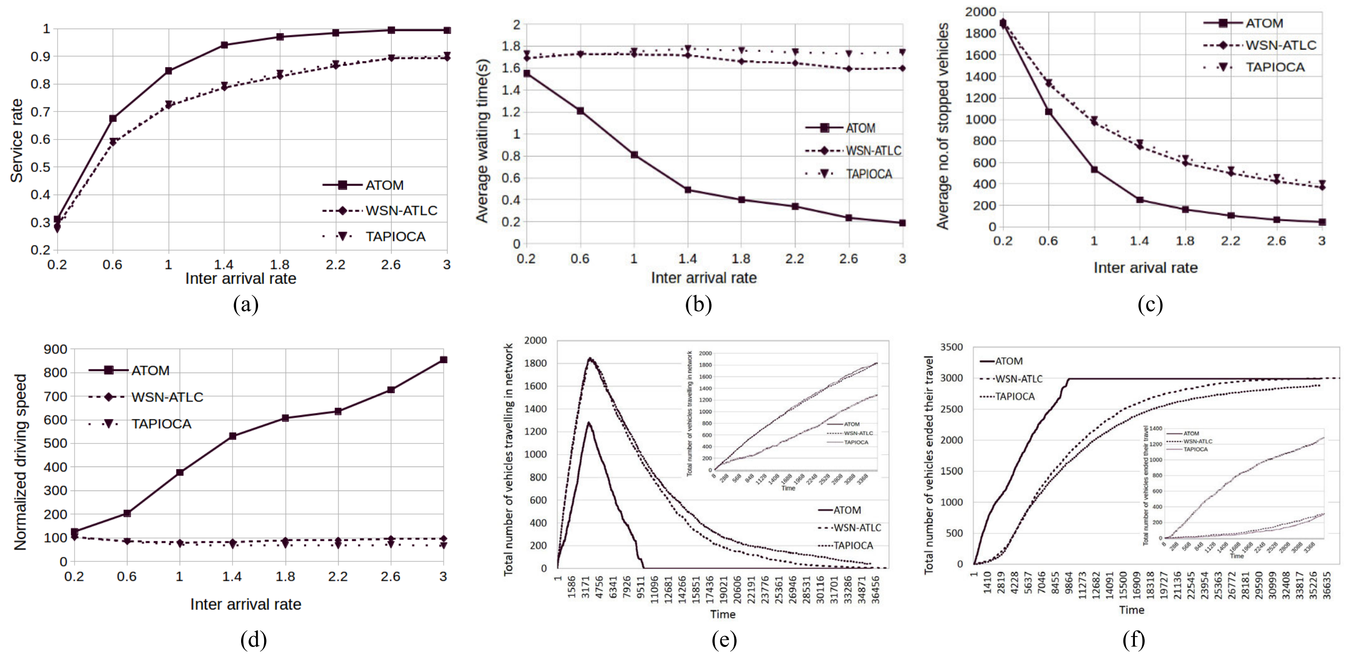


FIGURE 13. Performance comparison of ATOM to WSN-TLC, and TAPIOCA in a real road network in terms of various IARs for (a) service rate, (b) the average waiting time, (c) the average number of stopped vehicles for a green light and (d) normalized driving speed. (e) and (f) show performance of the approaches in terms of simulation time with IAR = 1.

road length in the setup are 2172 and 10470, respectively. For simulating the various road statuses, IAR is selected from 0.2 to 3 with an increment of 0.4.

Figures 13(a)-(d) show the effectiveness of ATOM compared to WSN-ATLC and TAPIOCA in terms of the service rate, the average waiting time, the average number of the

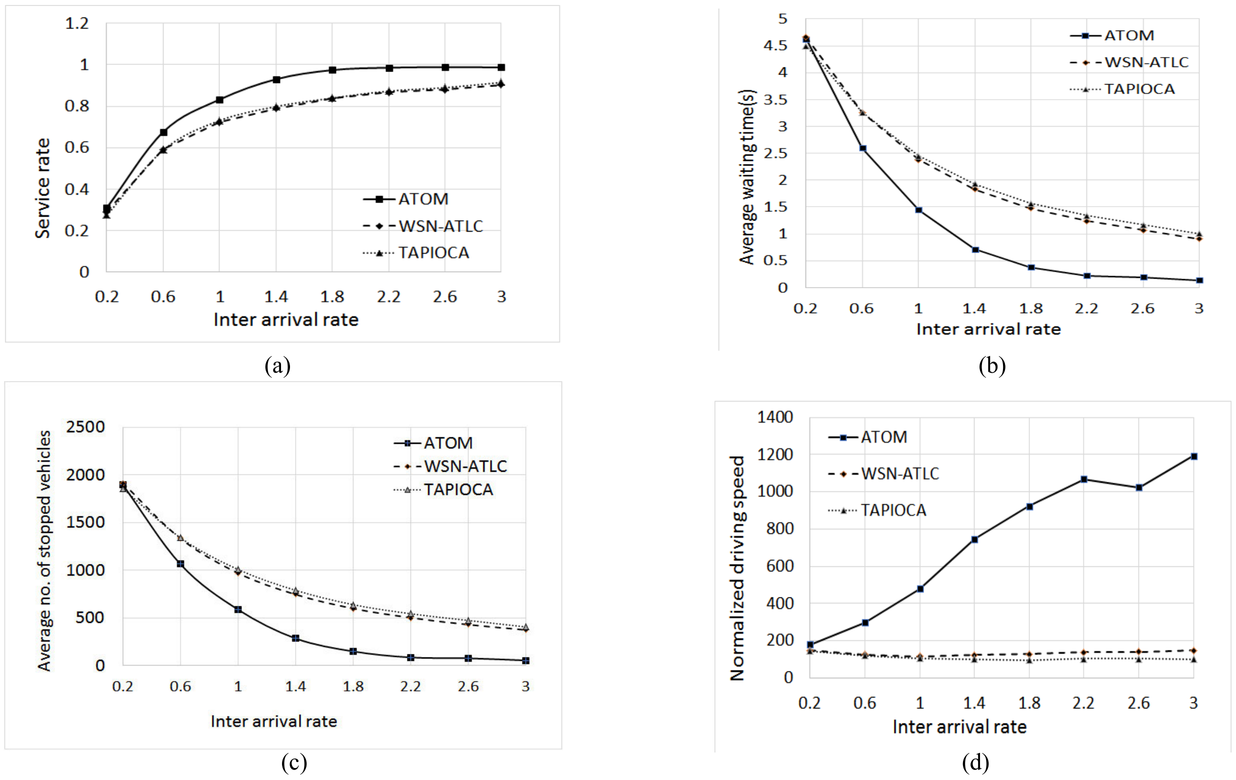


FIGURE 14. Performance comparison of ATOM to WSN-ATLC, FUZZY and TAPIOCA in a real road network in terms of various IARs using Logit route choice model; (a) service rate, (b) average waiting time, (c) average number of stopped vehicles and (d) normalized driving speed.

stopped vehicles for a green light and the average driving speed of vehicles during their trip. Figure 13(a) shows that ATOM serves more vehicles in the real road network than the alternatives with all IAR, which is consistent with the results of the grid networks. It is worth noting that ATOM reaches 100% service rate while the baselines cap around 90% at IAR = 2.6 to 3 when the road network is very much free. This is due to ATOM's adaptability to the various levels of road congestion. In addition, as the IAR decreases and the network gets more congested, ATOM keeps showing higher service rate than those of the baselines. ATOM's distinct performance is attributed to considering every possibility of the formed phases and computing durations using a flow-based traffic analysis; this particularly makes ATOM more adaptable to traffic fluctuation scenarios found in practice. Meanwhile, WSN-ATLC and TAPIOCA are not as flexible in selecting phases and computing green durations as ATOM.

Moreover, Figure 13(b)-(d) show that ATOM yields better performance than WSN-ATLC and TAPIOCA with respect to the average waiting time, the average number of stopped vehicles and the average driving speed. For a low congestion level (IAR = 3), Figure 13(b) shows that only about 6% of the driving vehicles need to stop for a green light at an intersection in ATOM, while about 40% of vehicles have to wait for a green light in WSN-ATLC and TAPIOCA. Such percentage reaches up to 50% when roads are very congested at IAR = 0.2, where total 4000 vehicles are added into the network. Moreover, it is worth to note that ATOM shows distinct improvement in the driving speed compared to the baselines, as shown in Figure 13(d). The enhancement can be again attributed to the mode selection and phase selection schemes of ATOM which help to reduce waiting time and consequently decreases the travel duration. WSN-ATLC and TAPIOCA do not yield much improvement in a driving speed even as IAR increases to three.

In addition, Figure 13(e) and (f) represent how fast and how much the roads are congested in each approach and how rapidly each algorithm can reach full service rate during 36000 second simulation time with IAR = 1. As shown in Figure 13(e) and (f), ATOM holds at most 43% of all inserted vehicles into the network and services all the loaded vehicles to finish their travel within less than one third of the entire simulation time. Meanwhile, in WSN-ATLC and TAPIOCA up to 60% of the vehicles occupy the network as seen in Figure 13(e) and Figure 13(f) shows they spend almost all simulation time (36000 seconds) to complete 3000 vehicles' travel. Overall ATOM can maintain the real road network under less congested status by relieving or preventing congestion on the road and thus serving vehicles more quickly than WSN-ATLC and TAPIOCA. Each of Figure 13(e) and (f) include a zoomed in plot to show the results during the first 3600 seconds in order to better compare the performance of WSN-ATLC and TAPIOCA.

Moreover, we have run the same experiments using Logit route choice model [35]. The results are shown in Figure 14. As indicated in Figure 14(b), WSN-ATLC and TAPIOCA experience rapid decline in the average waiting time as IAR grows than that of Figure 13(b). Meanwhile, the other charts of Figure 14 show the almost same results as those of Figure 13.

VI. CONCLUSION

In this paper we have presented ATOM, a novel traffic flow control algorithm. In ATOM each intersection estimates the volume of incoming traffic and also analyzes the congested status of the next road segment for each flow of vehicles. ATOM then adjusts traffic light phases and their duration in cooperation with neighboring intersections. In addition, ATOM can be applied to practice in a distributed or centralied way. ATOM is validated through extensive simulation experiments using both synthetic and real road network from the City of Baltimore. The simulation results have demonstrated that ATOM improves road throughput, average waiting time for a vehicle and the average number of vehicle stops in comparison to competing scheme. Basically, ATOM allows more vehicles to pass through the road network and decreases their travel time by helping more vehicles move even under high traffic congestion. As a major city, Baltimore experiences a wide variety of traffic patterns, e.g., rush hours, major sport events, parades, etc. Therefore, the distinct performance of ATOM in such setup under the various traffic congestion situations, confirms its effectiveness and is deemed as a sufficient evidence that ATOM can be invaluable in practice. Our future work includes the extension of the proposed algorithm considering pedestrians.

APPENDIX A

See Table 3.

TABLE 3. Acronyms and terminologies.

Acronyms	Description
$Len(S)$	Length of a road S
$CP(S)$	Full capacity of S
$dur_t(F_{ba})$	Green signal duration for F_{ba}
$TH_t(F_{ba})$	Number of vehicles of a flow F_{ba} which have departed from i_C during $dur_t(F_{ba})$
$NDV_t(F_{ba})$	Number of vehicles that got delayed at i_C, i.e., could not pass during CYC_{t-1}
$Delay_t(F_{ba})$	Average waiting time for the vehicles in F_{ba} waiting for a green light in CYC_t
$Speed_t(F_{ba})$	Average driving speed of vehicles $\in F_{ba}$
$NAV_t(F_{ba})$	Number of additional vehicles to an incoming flow F_{ba} towards i_C from its neighboring intersection i_C^a between the green signals in CYC_{t-1} and CYC_t.
$NEV_{t+1}(F_{ba})$	Expected number of vehicles joining F_{ba} which will head to i_C^b from i_C^a through i_C in the next cycle CYC_{t+1}.
CYC_t	Traffic signal cycle in which every F_{ba} is served at least once with a green signal.
ph	Traffic light phase, i.e., a combination of two or more flows, F_{ba}'s in non-conflicting directions and whose green lights can be
$dur_t(ph)$	Green period allowed for a phase ph in a cycle CYC_t.
$dur(CYC_t)$	Length of CYC_t which equals a sum of $dur_t(ph)$ for all phin the cycle,
$Speed_t(ph)$	Average speed of driving vehicles that belong to a phase ph in CYC_t.
$PH(CYC_t)$	List of the phases in CYC_t

APPENDIX B
PSEUDO CODE

See Algorithms 1–5.

Runtime Complexity of ATOM: ATOM operates at the traffic light controller of each intersection, as also illustrated in Figure 2. The runtime complexity of ATOM is mainly determined by four functions performed every cycle; these functions are TFW(), CMD() and TLSA-PS() and TLSA-DA().

- The complexity of TFW() is bounded to the number of flows that is a constant value (12) and the number of phases, i.e., | *PH(CYC$_t$)*| that is determined depending on a set of traffic light signals used at an intersection (supported directions). Thus TFW()'s complexity is bounded to *O(1)*.
- CMD() conducts only four comparisons and thus its complexity is *O(1)*.
- The main part of TLSA-PS() sorts the traffic signal phases in *PH(CYC$_t$)*. When using a merge or heap sorting algorithm, the complexity bounds to *O(nlogn)*, where n = | *PH(CYC$_t$)*|.
- The complexity of TLSA-DA() is bounded to *O(1)* since it includes five comparisons and simple numerical operations in the worst case.

Overall, when the time complexity of ATOM at each intersection is bounded to *O(nlogn)*, where n = |*PH(CYC$_t$)*|. The possible phases depend on the number of traffic light signals, which is bounded to twelve every cycle. The implementation of ATOM can be in a distributed manner, i.e., through collaborative execution at traffic controllers of local intersections, or logically centralized in a cloud server which has more computing power than a local controller. Therefore, it is feasible to apply ATOM to a real traffic network.

APPENDIX C
BASELINE APPROACHES

WSN-ATLC: Like ATOM, WSN-ATLC [3] schedules the green light sequences and durations based on the traffic information at local and nearby intersections. However, WSN-ATLC considers two lanes per road, $S(i_C^b i_C)$ and $S(i_C i_C^a)$, $ba \in NSWE$ each of which is dedicated for *GL* and *GS* only. Thus, it does not control *GR* traffic and has a smaller set ALL'_{ph} of phases than ATOM by including only the twelve phases out of all combinations seen in Figure 4(a). WSN-ATLC consists of two steps; one is to determine which phase should obtain a green signal next and the other is to optimize

Algorithm 1 Pseudo code of monitoring & evaluation by TFW

```
TFW() {
    // Update flow-based evaluation parameters
1.  for $\forall F_{ba} \in S_{in\_flow}$ at $i_C$ {
2.      Update $Delay_t(F_{ba})$, $NDV_t(F_{ba})$, $TH_t(F_{ba})$,
        $Speed_t(F_{ba})$, and $NAV_t(F_{ba})$;
3.      $FOR_t(F_{ba}) \leftarrow \frac{Delay_t(F_{ba})}{FreeTimeDelay} \times \frac{(NDV_t(F_{ba}) + NAV_t(F_{ba}))}{CP(S(i_C^b, i_C))}$;
4.      $CL_t(F_{ba}) \leftarrow \frac{FOR_t(F_{ba})}{\sum_{\forall F \in i_C} FOR_t(F)} + \frac{TH_{t-1}(i_C)}{TH_{t-1}(F_{ba})} + \frac{NEV_t(F_{ba})}{NEV_t(i_C)}$;
5.  } end for
6.  for $\forall ph \in PH(CYC_t)$ at $i_C$ {// Update phase-based ones
7.      $CL_t(ph) \leftarrow \sum_{\forall F_{ba} \in ph} CL_t(F_{ba})$;
8.  } end of for
    // Update the average values at $i_C$;
9.  $Delay_t(i_C) = (\sum_{\forall F_{ba} \in i_C} Delay_t(F_{ba}))/N_{in\_flow}$;
10. $TH_t(i_C) = (\sum_{\forall F_{ba} \in i_C} TH_t(F_{ba}))/N_{in\_flow}$;
11. $NDV_t(i_C) = (\sum_{\forall F_{ba} \in i_C} NDV_t(F_{ba}))/N_{in\_flow}$;
12. Re-compute $CL_t(i_C)$;
}
```

Algorithm 2 Pseudo code of congestion decision by TFW

```
CMD(){
1.  if( $FOR_t(i_C) \leq P_{FFP}$ ) then{
2.      if ( $Mode_t(i_C) \neq FFP$ ) then {
3.          $Mode_t(i_C) \leftarrow$ the 1st level of $FFP$;
            // switch to $FFP$
4.      } else if( $TH_{t-1}(i_C) \geq TH_t(i_C)$
            && $Delay_{t-1}(i_C) \leq Delay_t(i_C)$) then {
5.          if ( $Mode_t(i_C)$ is the highest level of $FFP$ )
            then {
6.              $Mode_t(i_C) \leftarrow CAP$;
7.              $P_{FFP} \leftarrow FOR_t(i_C)$; // threshold value is
                adjusted!
8.          }else{
9.              $Mode_t(i_C) \leftarrow$ the next level of $FFP$; }
10.     } end if
11. } else{
12.     switch ( $Mode_t(i_C)$)
13.         case $FFP$:
14.             if ( $FOR_t(i_C) > P_{FFP}$ ) then{
15.                 $Mode_t(i_C) \leftarrow$ ready-to-leave; }
16.             break;
17.         case ready-to-leave:
18.             if (($FOR_t(i_C) > P_{FFP}$ &&
                ($TH_{t-1}(i_C) \geq TH_t(i_C)$
                &&$Delay_{t-1}(i_C) \leq Delay_t(i_C)$))
                || ( $\exists F_{ba} FOR_t(F_{ba}) > P_{FFP}$ )) then{
19.                 $Mode_t(i_C) \leftarrow CAP$;
20.             } end if
21.             break;
22.         case $CAP$:
23.             if ( $FOR_t(i_C) > P_{FFP}$ &&
                $FOR_t(F_{ba}) \leq P_{FFP}, \forall F_{ba}$) then{
24.                 $Mode_t(i_C) \leftarrow$ ready-to-leave;
25.             break;
26.     } end switch
27. } end if
}
```

the green length of the selected phase. The optimization factors in the *traffic volume*, *waiting time*, and *the number of stops* in each phase $\in ALL'_{ph}$, *hunger level* that denotes the starvation of lanes for green light, *blank circumstance* which reflects blank spaces on lanes and *special circumstance* that refers to some situations where a green or red light must be activated urgently.

WSN-ATLC assigns a green light to the phase which has no blank lanes, i.e., fully occupied lane with the highest priority. Otherwise, the phase that has the highest value of the weighted sum of the six factors takes a turn for a green. As a result, WSN-ATLC does not consider traffic information obtained from neighboring intersections $i_C^x, x \in NSWE$ during phase selection. The traffic status at i_C^x's is only factored in determining the length of green light duration. The computed green time is picked such that all waiting vehicles at i_C and the expected number of additional vehicles coming from neighbor intersections towards i_C are allowed to pass. Unlike ATOM, WSN-ATLC does not consider the congestion level of the road segments that vehicles drive into after passing i_C.

FUZZY: In order to monitor real-time traffic status, FUZZY employs multiple fuzzy logic controllers, interconnected using IEEE 802.15.4 technology [19]. Like ATOM, FUZZY dynamically orders phases and calculates green time while factoring turning. However, unlike ATOM it does not consider per-flow traffic status to determine the best combination of phases in order to increase traffic throughput. In other words, FUZZY organizes a fixed set of four phases using only the two traffic signal lights, {GL only, GS&GR} among all possible combinations seen in Figure 4(a). FUZZY uses using road sensors; depending on the number of queued vehicles in the lane, it classifies each lane in a phase into three levels, namely, normal, medium and long. Then the four phases are scheduled according to a sum of the assigned levels to the lanes belonging to each phase. As a result,

Algorithm 3 Pseudo code of TLSA-PS() by TLSA

```
TLSA-PS() {
1.  switch ( $Mode_t\ (i_C)$ ) {
2.     case 1^st sub-level of $FFP$ :
3.        return;
4.     case 1^st or 2^nd sub-level of $FFP$:
5.        for each $ph \in PH(CYC_t)$ {// Compute a priority of $ph$
6.           $CL_t\ (ph) \leftarrow \sum_{\forall F_{ba} \in ph} CL_t\ (F_{ba})$;
7.        } end for
8.        Sort $PH(CYC_t)$ in descending order of $CL_t\ (ph)$;
9.        return $PH(CYC_t)$;
10.    case $CAP$ : // Phase selector as well as sorter
11.       $\leftarrow PH(CYC_t)\emptyset; \leftarrow Tmp_F S_{in\_flow}; x \leftarrow 0$;
12.       $Tmp_F - = \{\forall F_{ba}, dur_{t+1}\ (F_{ba}) == 0 | F_{ba} \in ph\}$ ;
13.       do{
14.          $f_1 \leftarrow F_{ba}$ where $CL_t(F_{ba})$;
15.          for $\forall ph \in ALL_{ph}$ which includes $f_1$ {
16.             $ph \leftarrow ph$ in which the deviation of
                $dur_{t+1}\ (F_{sd})\ , \forall F_{ba} \in ph$ is the least;
17.          } end for
18.          $PH(CYC_t) \cup = \{ph\}$ in $x$-th place;
19.          $dur_{t+1}\ (ph) = (dur_{t+1}\ (F_{ba}))$;
20.          $Tmp_F - = \{\forall F_{ba} | F_{ba} \in ph\}\ ; x + +$;
21.       }while ( $\exists F_{ba} \notin PH(CYC_t)$ or $Tmp_F \neq \emptyset$ )
22.       return $PH(CYC_t)$ and $dur_{t+1}(ph)$;
23. }end switch
}
```

Algorithm 4 Pseudo code of TLSA-DA() by TLSA

```
TLSA-DA() {
1.  switch ( $Mode_t\ (i_C)$ ) {
2.     case 1^st or 2^nd sub-level of $FFP$:
3.        break;
4.     case 3^rd sub-level of $FFP$:
5.        $dur_{t+1}\ (ph)\ , \forall ph \leftarrow \sum_{\forall F_{ba} \in ph} \frac{(NDV_t(F_{ba}) + NAV_t(F_{ba}) + NEV_{t+1}(F_{ba})) \times Len(S)}{Speed_t(F_{ba})}$;
6.        $dur\ (CYC_{t+1}) \leftarrow \sum_{ph \in S_{ph}} dur_{t+1}\ (ph)$;
7.        return $dur_{t+1}(ph)$, $\forall ph$ and $dur\ (CYC_{t+1})$;
8.     case $CAP$:
9.        for each $F_{ba} \in S_{in\_flow}$ {
10.          $\gamma_{TH} = \frac{TH_t(F_{ba})}{TH_{t-1}(F_{ba})}; \gamma_{DELAY} = \frac{Delay_t(F_{ba})}{Delay_{t-1}(F_{ba})}$;
11.          $\gamma_{NDV} = \frac{NDV_t(F_{ba})}{NDV_{t-1}(F_{ba})}; \gamma_{NVOL} = \frac{NEV_{t+1}(F_{ba})}{NEV_t(F_{ba})}$;
12.          if ( $\gamma_{TH} > 1$ ) then{// case1 or 2
13.             if ( $\gamma_{DELAY} \leq 1$ ) then{
14.                if ( $\gamma_{NDV} \leq 1$ ) then { $CR_t = 1$; } // c1-1
15.                else{ $CR_t = \gamma_{NDV}$; } // c1-2 }
16.             } else {// c2
17.                if ( $\gamma_{NDV} \leq 1$ ) then { $CR_t = 1$; } // c2-1
18.                else{ $CR_t = \gamma_{NDV}$; } // c2-2
19.             }end if
20.          } else if ( $\gamma_{TH} \leq 1$ ) then{// case3 or 4
21.             if ( $\gamma_{DELAY} \leq 1$ ) then{
22.                if ( $\gamma_{NDV} \leq 1$ ) then {
23.                   if ( $\gamma_{NVOL} > 1$ ) then { $CR_t = 1$; } // c3-1
24.                   else{ $CR_t = \gamma_{DELAY}$; } // c3-2
25.                }
26.             } else {// c4
27.                if ( $\gamma_{NDV} > 1$) then { $CR_t = 1$; } // c4-1
28.                else{ $CR_t = \gamma_{TH}$; } // c4-2
29.             }end if
30.          } end if
      // removing inefficient green signals without starvation
31.          if ( $dur_t\ (F_{ba}) \times CR_t >$ MDT) then{
32             $dur_{t+1}\ (F_{ba}) = dur_t\ (F_{ba}) \times CR_t$;
33.          } else{// no green light for $F_{ba}$ in $CYC_{t+1}$
34.             if ( $Starvation(F_{ba}) > TH_{starvation}$ ) then{
35.                $Starvation(F_{ba}) = 0$; }
36.             else { $dur_{t+1}\ (F_{ba}) = 0$; $Starvation(F_{ba}) + +$;
}
37.       } end for
38.       return $dur_{t+1}(F_{ba})$, $\forall sd$;
39. } end switch
}
```

the phase with more queued vehicles takes higher priority to obtain a green light in the next cycle. The green duration is calculated according to the queue length of vehicles in the lanes belonging to the phase.

Since FUZZY tries to control vehicular traffic per phase without considering traffic status of the individual flows in a phase, flow-level traffic management cannot be achieved unlike ATOM. In addition, FUZZY manages traffic lights at i_C by only relying on the current traffic condition on the roads towards i_C without factoring in the expected volume of the incoming vehicles or the outgoing road conditions.

TAPIOCA: It exploits vehicular traffic data collected by a WSN and dynamically decides the green light sequences and durations through collaboration among multiple intersections, e.g., using I2I communication [4]. TAPIOCA denotes the different traffic volume between incoming and outing roads as *the expected gain* for a movement. TAPIOCA first computes *a local movement score* of vehicles passing i_C by considering the number and delay of vehicles per road from direction b to a at i_C, where $ba \in \{N, S, W, E\}$. Like ATOM, TAPIOCA considers traffic congestion and tries to slow down vehicles that are moving towards congested outgoing roads $S\left(i_C, i_C^a\right)$. However, TAPIOCA assesses the outgoing road congestion by computing the difference of vehicles between $S\left(i_C, i_C^a\right)$ and $S\left(i_C^b, i_C\right)$. This is different from ATOM which estimates the traffic condition on $S\left(i_C, i_C^a\right)$ by considering changes of passing traffic volume, waiting time, delayed vehicles and incoming vehicles per flow towards the road. Therefore, TAPIOCA does not suit setups in which road capacity varies.

After that TAPIOCA computes a global score for each road at i_C from direction b to a, which is equal to a sum of its

Algorithm 5 Pseudo code of ATOM

```
Initialize(){
1. $PH(CYC_t) \leftarrow INI_{ph}$; // Initialize a set of phases
2. $dur(CYC_t) \leftarrow INI_{CYC}$; // short initial value, 120 sec
3. $dur_t(F_{ba}), \forall F_{ba} \leftarrow INI_{dur}$; // = $\frac{dur(CYC)}{|PH(CYC_t)|}$
4. $Mode_t\ (i_C) \leftarrow 1^{st}$ sub-level of $FFP$;
5. MDT $\leftarrow 5$; // minimum duration threshold used to
remove useless green duration (sec) less than the required
time for a vehicle to pass $i_C$;
6. $TH_{starvation} \leftarrow 3$; $\forall F_{ba}$, $Starvation(F_{ba}) = 0$;
}
ATOM(){
1.  Initialize();
2.  do( ) {
3.      if ( each time $CYC_t$ ends ) then {
4.        TFW(); // Update traffic status
5.        $CMD$(); // Update congestion mode
6.        switch ( $Mode_t\ (i_C)$ ) {
7.          case $1^{st}$ or $2^{nd}$ sub-level of $FFP$:
8.              $dur(CYC_{t+1}) \leftarrow Delay_t\ (i_C) \times |PH(CYC_t)|$;
9.              break;
10.         case $2^{nd}$ or $3^{rd}$ sub-level of $FFP$:
11.             $PH(CYC_t) \leftarrow$ TLSA-PS ( $Mode_t\ (i_C)$ );
12.             $dur(CYC_{t+1}) \leftarrow \sum_{\forall ph \in PH(CYC_t)} dur_{t+1}\ (ph)$;
13.             break;
14.         case $3^{rd}$ sub-level of $FFP$:
15.             $dur_{t+1}(ph), \forall ph \leftarrow$ TLSA-DA( $Mode_t\ (i_C)$);
16.             $dur(CYC_{t+1}) \leftarrow \sum_{\forall ph \in PH(CYC_t)} dur_{t+1}\ (ph)$;
17.             break;
18.         case $CAP$:
19.             $dur_{t+1}(F_{ba}), \forall F_{ba} \leftarrow$ TLSA-DA ( $CAP$ );
20.             $PH(CYC_t)$ and $dur_{t+1}(ph)$ $\leftarrow$ TLSA-PS(
$CAP$ );
21.             $dur(CYC_{t+1}) \leftarrow \sum_{\forall ph \in PH(CYC_t)} dur_{t+1}\ (ph)$;
22.             break;
23.         } end switch
24.     } end if // each time $CYC_t$ ends
        // when neighboring traffic info arrives at $i_C$
25.     if ( any data received from $i_C^x$, $x = \{NSWE\}$ ) then
{
26.         $NEV_{t+1}\ (F_{xa}) \leftarrow \sum_{x \neq a \in \{N,S,E,W\}} TH_t\ (F_{xa})$;
27.     }end if
28.
}
```

local movement score, *expected gain* and *rank*. The rank is used for creating green waves and its value corresponds to a global score of b computed by a neighboring intersection of i_C, i.e., i_C^b in a direction b. In other words, the rank value is computed by ordering neighboring intersections, i_C^b's in a decreasing order of the summed score of the roads heading to b from i_C. Based on the global score, TAPIOCA selects a phase by combining non-conflict movements using conflict matrix that make a high score and schedules them in a decreasing order of the scores. Then, it calculates green duration of each phase that suffices for all waiting vehicles to cross i_C. Subsequently, the global scores at i_C are shared with its neighboring intersections i_C^b to compute a global score at i_C^b.

TAPIOCA computes green duration based on the maximum number of waiting traffic volume, limits it to a certain threshold and does not consider the outbound road status, as well as changed road condition. Thus, TAPIOCA provides unneeded green time. Additionally, like WSN-ATLC, TAPIOCA has restriction on the number of lanes which form a phase to two lanes at all times, unlike ATOM which uses all possible combinations of lanes during the phase selection. Similar to WSN-ATLC and FUZZY, TAPIOCA monitors and assess traffic status only per lane, unlike ATOM which does per flow as well as per lane, which provides better assessment of the road condition.

REFERENCES

[1] H. Hartenstein and K. Laberteaux, *VANET Vehicular Applications and Inter-Networking Technologies*. Hoboken, NJ, USA: Wiley, 2010.

[2] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system," in *Proc. IEEE 72nd Veh. Technol. Conf.*, Ottawa, ON, Canada, Sep. 2010, pp. 1–5.

[3] B. Zhou, J. Cao, and H. Wu, "Adaptive traffic light control of multiple intersections in wsn-based ITS," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC Spring)*, Yokohama, Japan, May 2011, pp. 1–5.

[4] S. Faye, C. Chaudet, and I. Demeure, "A distributed algorithm for multiple intersections adaptive traffic lights control using a wireless sensor networks," in *Proc. 1st Workshop Urban Netw.* New York, NY, USA: ACM, 2012, pp. 13–18.

[5] M. Milojevic and V. Rakocevic, "Distributed road traffic congestion quantification using cooperative VANETs," in *Proc. 13th Annu. Medit. Ad Hoc Netw. Workshop (MED-HOC-NET)*, Piran, Slovenia, Jun. 2014, pp. 203–210.

[6] F. Soylemezgiller, M. Kuscu, and D. Kilinc, "A traffic congestion avoidance algorithm with dynamic road pricing for smart cities," in *Proc. IEEE 24th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, London, U.K., Sep. 2013, pp. 2571–2575.

[7] J. Pan, I. S. Popa, K. Zeitouni, and C. Borcea, "Proactive vehicular traffic rerouting for lower travel time," *IEEE Trans. Veh. Technol*, vol. 62, no. 8, pp. 3551–3568, Oct. 2013.

[8] S. R. Azimi, G. Bhatia, and R. R. Rajkumar, "Reliable intersection protocols using vehicular networks," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, 2013, pp. 1–10.

[9] R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige, "STIP: Spatio-temporal intersection protocols for autonomous vehicles," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2014, pp. 1–12.

[10] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *Proc. Amer. Control Conf.*, Chicago, IL, USA, Jul. 2015, pp. 763–768.

[11] F. Ahmad, S. A. Mahmud, G. M. Khan, and F. Z. Yousaf, "Shortest remaining processing time based schedulers for reduction of traffic congestion," in *Proc. Int. Conf. Connected Vehicles Expo (ICCVE)*, Las Vegas, NV, USA, Dec. 2013, pp. 271–276.

[12] K. Pandit, D. Ghosal, H. M. Zhang, and C. N. Chuah, "Adaptive traffic signal control with vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1459–1471, May 2013.

[13] C. Li and S. Shimamoto, "An open traffic light control model for reducing vehicles' CO_2 emissions based on ETC vehicles," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 97–110, Jan. 2012.

[14] S. Kwatirayo, J. Almhana, and Z. Liu, "Adaptive traffic light control using VANET: A case study," in *Proc. 9th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jul. 2013, pp. 752–757.

[15] C. Hu and Y. Wang, "A novel intelligent traffic light control scheme," in *Proc. 9th Int. Conf. Grid Cloud Comput. (GCC)*, pp. 372–376, 2010.

[16] S. Sameh, A. El-Mahdy, and Y. Wada, "A linear time and space algorithm for optimal traffic-signal duration at an intersection," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 387–395, Feb. 2015.

[17] H. Prothmann *et al.*, "Organic traffic light control for urban road networks," *Int. J. Auton. Adapt. Commun. Syst.*, vol. 2, no. 3, pp. 203–225, 2009.

[18] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck, "Possibilities and limitations of decentralised traffic control systems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, Jul. 2010, pp. 1–9.

[19] M. Collotta, L. L. Bello, and G. Pau, "A novel approach for dynamic traffic lights management based on Wireless Sensor Networks and multiple fuzzy logic controllers," *Expert Syst. Appl.*, vol. 42, pp. 5403–5415, Aug. 2015.

[20] M. Elgarej, M. Khalifa, and M. Youssfi, "Traffic lights optimization with distributed ant colony optimization based on multi-agent system," in *Proc. Int. Conf. Networked Syst.* Springer, 2016, pp. 266–279.

[21] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown Toronto," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, Sep. 2013.

[22] I. Dusparic and V. Cahill, "Autonomic multi-policy optimization in pervasive systems: Overview and evaluation," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 1, p. 11, 2012.

[23] A. L. C. Bazzan, D. de Oliveira, and B. C. da Silva, "Learning in groups of traffic signals," *Eng. Appl. Artif. Intell.*, vol. 23, no. 4, pp. 560–568, 2010.

[24] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," *Eng. Appl. Artif. Intell.*, vol. 26, nos. 5–6, pp. 1575–1587, 2013.

[25] X.-F. Xie, S. F. Smith, and G. J. Barlow, "Schedule-driven coordination for real-time traffic network control," in *Proc. 22nd Int. Conf. Automated Planning Scheduling (ICAPS)*, 2012, pp. 323–331.

[26] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Proc. 16th Annu. Symp. Found. Comput. Sci.*, 1975, pp. 184–193.

[27] M. Hausknecht, T.-C. Au, P. Stone, D. Fajardo, and T. Waller, "Dynamic lane reversal in traffic management," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 1929–1934.

[28] T. G. Crainic, N. Ricciardi, and G. Storchi, "Advanced freight transportation systems for congested urban areas," *Transp. Res. C, Emerg. Technol.*, vol. 12, no. 2, pp. 119–137, 2004.

[29] Y. Iida, "Basic concepts and future directions of road network reliability analysis," *J. Adv. Transp.*, vol. 33, no. 2, pp. 125–134, 1999.

[30] V. Vukadinovic *et al.*, "3GPP C-V2X and IEEE 802.11p for Vehicle-to-Vehicle communications in highway platooning scenarios," *Ad Hoc Netw.*, vol. 74, pp. 17–29, May 2018.

[31] Q. Wang, J. Zheng, H. Xu, B. Xu, and R. Chen, "Roadside magnetic sensor system for vehicle detection in urban environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1365–1374, May 2018.

[32] X. Yang, L. Tang, K. Stewart, Z. Dong, X. Zhang, and Q. Li, "Automatic change detection in lane-level road networks using GPS trajectories," *Int. J. Geograph. Inf. Sci.* vol. 32, no. 3, pp. 601–621, 2018.

[33] M. Hsu and A. Shih, "A traffic signal control mechanism in a connected vehicle environment," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan*, Taipei, Taiwan, Jun. 2015, pp. 256–257.

[34] *DLR—Institute of Transportation Systems—SUMO—Simulation of Urban Mobility*. Accessed: Nov. 11, 2016. [Online]. Available: https://www.dlr.de/dlr/desktopdefault.aspx/tabid-10002/

[35] E. Cascetta, A. Nuzzolo, F. Russo, and A. Vitetta, "A modified logit route choice model overcoming path overlapping problems. Specification and some calibration results for interurban networks," in *Proc. ISTTT Conf.*, Lyon, France, 1996, pp. 697–711.

[36] N. Bhouri, F. J. Mayorano, P. A. Lotito, H. H. Salem, and J. P. Lebacque, "Public transport priority for Multimodal urban traffic control," *Cybern. Inf. Technol.*, vol. 15, no. 5, pp. 17–36, 2015.

[37] *How SCATS Works | SCATS*. Accessed: Dec. 30, 2018. [Online]. Available: https://www.scats.com.au/how-scats-works.html

[38] *Search Results—DPTI—Department of Planning, Transport and Infrastructure South Australia*. Accessed: Dec. 30, 2018. [Online]. Available: https://www.dpti.sa.gov.au/__data/assets/pdf_file/0017/151172/TRAFFIC_MODELLING_GUIDELINES_-_TRANSYT_15_PUBLICATION_VERSION_CREATED_11_11_2014_16_03.pdf

[39] *TRL Software | TRANSYT—TRL Software*. Accessed: Dec. 30, 2018. [Online]. Available: https://trlsoftware.com/products/junction-signal-design/transyt/

SOOKYOUNG LEE received the B.S. and M.S. degrees in computer science from Ewha Womans University, South Korea, in 1995 and 1997, respectively, and the Ph.D. degree in computer science from the University of Maryland, Baltimore County, USA, in 2010. She has been with LG Electronics, Inc., Electronics and Telecommunications Research Institute, Korea Electrics Technology Institute, and Samsung Electronics Co., Ltd., South Korea, from 1998 to 2004. While at LG, she has developed the IP data server over ATM switch and implemented virtual private network service for multiprotocol label switching systems. She was a Volunteer of IPv6 forum Korea while at ETRI and has developed the network address and protocol translation systems between IPv4 and IPv6. At Samsung, she was a broadband convergence network designer especially focusing on requirements for QoS and IPv6. She is currently a Research Professor with the Department of Computer Science and Engineering, Ewha Womans University. Her current research interests include network architectures and protocols, topology restoration and fault tolerance in wireless sensor networks, network modeling, and performance analysis for dynamic and sparse ad-hoc networks.

MOHAMED YOUNIS (SM'96) received the Ph.D. degree in computer science from the New Jersey Institute of Technology, USA. He was with the Advanced Systems Technology Group, an Aerospace Electronic Systems R&D Organization of Honeywell International, Inc. While at Honeywell, he led multiple projects for building integrated fault tolerant avionics and dependable computing infrastructure. He also participated in the development of the redundancy management systems, which is a key component of the vehicle and mission computer for NASA's X-33 space launch vehicle. He is currently an Associate Professor with the Department of Computer Science and Electrical Engineering, University of Maryland at Baltimore County (UMBC), Baltimore. He has published more than 240 technical papers in refereed conferences and journals. He has seven granted and three pending patents. His current interest includes network architectures and protocols, wireless sensor networks, embedded systems, fault tolerant computing, secure communication, and distributed real-time systems. In addition, he serves/served on the editorial board of multiple journals and the organizing and technical program committees of numerous conferences. He is a Senior Member of the IEEE Communications Society.

AISWARYA MURALI received the B.Tech. degree in electronics and communication engineering from Calicut University, Kerala, India. She is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, Ewha Womans University. Her research interests include traffic light control algorithms and driverless vehicle control algorithms.

MEEJEONG LEE received the B.S. degree in computer science from Ewha Womans University, Seoul, South Korea, in 1987, the M.S. degree in computer science from the University of North Carolina, Chapel Hill, in 1989, and the Ph.D. degree in computer engineering from North Carolina State University, Raleigh, in 1994. In 1994, she joined the Department of Computer Science and Engineering, Ewha Womans University, where she is currently a Professor. She has been engaged in research in computer communications and networks, and performance modeling and evaluation. Her current research interests include protocols and architectures of mobile and wireless networks and the future Internet.