Autonomous Agents and Multi-Agent Systems (2020) 34:17
https://doi.org/10.1007/s10458-019-09434-w

A collaborative agent-based traffic signal system for highly dynamic traffic conditions

Behnam Torabi[1] · **Rym Z. Wenkstern**[1] · **Robert Saylor**[2]

Abstract
In this paper we present DALI, a distributed, collaborative multi-agent traffic signal timing system (TST) for highly dynamic traffic conditions. In DALI, intersection controllers are augmented with software agents which collaboratively adapt signal timings by considering the feedback of all controller agents that may be affected by a change. The model is based on a real-world TST and will be deployed with minimal changes to the infrastructure. DALI has been validated by traffic engineers as well as through extensive simulation of the City of Richardson's traffic network, comprising 128 signalized intersections. The experimental results show that, in highly dynamic scenarios, DALI outperforms the conventional traffic system used by the city as well as a state-of-the-art reinforcement learning-based TST.

Keywords Intelligent transportation system · Multi-agent systems · Coordinated decision making

1 Introduction

Traffic signals impact virtually everyone, every day. Whether on congested or uncongested routes, traffic signals punctuate every urban trip and have a direct impact on drivers, the environment, and the economy.

Several Traffic Signal Timing systems (TST) have been proposed by manufacturers, traffic engineers and researchers. The purpose of a TST is to coordinate individual traffic signals to achieve network-wide operational objectives. A TST usually consists of several components: (a) a number of intersection controllers, i.e., devices which control the operation of the

✉ Rym Z. Wenkstern
rymw@utdallas.edu

Behnam Torabi
behnam.torabi@utdallas.edu

Robert Saylor
robert.saylor@cor.gov

[1] Department of Computer Science, University of Texas at Dallas, Richardson, USA

[2] Department of Transportation and Mobility, City of Richardson, TX, USA

Published online: 13 January 2020

intersection's traffic signals; (b) a communication network and (c) either a central computer or network of computers to manage the system. Coordination between the controllers is either implicit (e.g., time-based) or explicit (e.g., through communication links).

Generally, real-world TSTs define the traffic signal timing problem as the search for optimal values for a set of signal timing parameters (e.g., split, cycle length, offset) that minimize or maximize an objective function (e.g., minimizing delay, minimizing travel time, maximizing traffic flow). Many TSTs have been deployed and used for decades. Traditional TSTs are controlled by a central computer [40,41] which allows for the efficient coordination of intersection controllers under normal traffic conditions. Unfortunately, these systems do not perform well when unexpected traffic disruptions occur. Modern TSTs assign the optimization calculations to the intersection controllers. Coordination between these controllers is implicit and involves only direct neighbors. To ensure broader-level coordination, most systems make use of a higher-level computer which explicitly communicates constraints to the controllers.

From a research perspective, the application of the agent paradigm to traffic signal timing has been of interest to researchers for some time. Distribution, autonomy and coordination are agent properties that are naturally suited for the traffic domain. In the context of traffic signal timing, researchers have proposed the use of a variety of techniques (e.g., game theory [5,9], neural networks [8,34], fuzzy logic [7,11]), and the commonly used reinforcement learning [6,15]. Most agent-based models, architectures and methods are academic. As such, they often simplify the signal timing problem and are validated on simple simulated traffic networks. The very few agent-based systems that were validated on models of real cities using real-world data [14,16] are based on assumptions that cannot be implemented in a real-world setting.

In this paper we present DALI (Distributed Agent-based traffic Lights) a multi-agent coordinated TST for congestion reduction. In DALI [38,39], each intersection controller is augmented with a software agent which continuously monitors the state of the intersection. When congestion is detected, an agent does not make decisions in isolation but consults with all intersection agents that may be affected by a change and considers their feedback in the definition of a new timing plan. This non-selfish, collaborative approach benefits all intersections potentially concerned by congestion.

The system will be deployed in the City of Richardson, Texas, with minimal changes to the traffic control infrastructure. Our work differs from existing solutions in that: (1) it is directly implementable in an existing TST; (2) it is fully decentralized and coordination between controllers is explicit, bidirectional, and involves all controllers affected by a change; and 3) it has been validated by traffic engineers as well as through an agent-based simulation of a real City, with the largest road network simulated so far.

We implemented DALI in MATISSE 3.0 [3,36,37], a large-scale multi-agent traffic simulation system, and interfaced a deployed controller with the simulator. Experiments were run on a model of the City of Richardson comprising its 128 existing intersections. This model is a replica of the City's road network and considers the exact road structure; turn and through lanes; local, collector and service roads; and detectors. We used real-world data provided by the City of Richardson to run various experiments. The results show that our agent-based system outperforms the conventional traffic system used by the city as well as a state-of-the-art reinforcement learning-based TST.

The remainder of this paper is organized as follows: Sect. 2 reviews existing works. Section 3 discusses the agent algorithms and Sect. 4 discusses the experimental results.

2 Related work

The traffic signal timing problem has been traditionally formulated as an optimization problem, i.e., finding the optimal (or near-optimal) values for a set of signal timing parameters with the goal of minimizing an objective function (e.g., vehicle travel time, delay). A plethora of optimization techniques have been discussed in the literature [26]. In addition to the traditional optimal-setting-search-based methods, AI techniques such as game theory [5,9], neural networks [8,34], fuzzy logic [7,11], and reinforcement learning [2,6,15,16] have been used to propose solutions to the signal timing problem.

In the remainder of this paper, we restrict our discussion to TSTs that have been successfully deployed and used, as well as applied academic works that have been validated on realistic models, with real-world data.

In TSTs, signal timing optimizations are computed at the network/sub-network level or at the intersection level.

2.1 Network or sub-network level

TSTs in this category include the widely-used TRANSYT [41], SCOOT [40] and SCATS [31], as well as TUC [12]. These systems are centralized, i.e., controllers are managed by a central or several regional computers whose roles are to select the appropriate signal plans. Traffic data is either collected over time then processed (off-line system), or passed onto a computer in real-time (online system).

TRANSYT is an off-line system which uses historical data to calculate the network's performance index and then applies an optimization process to determine whether changes to the signal settings will improve the index. The main limitation of TRANSYT is the use of historical data which often results in timing plans that are out-of-date and ill-matched to the current traffic conditions.

SCOOT is an online TST. Traffic data is collected in real-time and processed every seconds. The data is passed on to a central computer which computes the *cycle flow profile* used to predict queue lengths. The predictions are passed onto an optimizer which determines the best pre-defined plan to reduce the likelihood of congestion (i.e., a queue blocking the upstream junction).

SCATS [31] is one of the most widely used TST in the world. SCATS is structured as a three-layered hierarchical system with a control center at the highest level, followed by regional computers in the next layer and local intersection controllers at the lower layer. The central computer monitors the system performance whereas the regional computers execute area-based adaptive strategies. When coordination is deemed necessary, a regional master requests that a set of local controllers execute a pre-defined coordinated plan. Changes to pre-defined plans are done manually by traffic engineers.

TRANSYT, SCOOT and SCATS were primarily designed to respond to time-of-day and long-term variations in traffic. Their strategy is based on increasing the timings at intersections and does not account for shortening or skipping a phase. In addition, SCOOT and SCATS make use of real-time measurements from the intersections incoming roads only. As such these systems are not adequate to deal with unexpected traffic disruptions.

TUC [12] is a recent centralized TST which formulates the traffic control problem as a Linear-Quadratic optimal control problem. TUC considers all traffic intersections simultaneously through the application of a single matrix equation. Results show that TUC is able to achieve highly efficient and extremely simple coordinated control strategies in large traffic

networks. Although the system was deployed in the Glasgow area and has proven to be efficient, its centralized architecture requires that the strategy be completely re-designed (i.e., all control matrices be re-calculated) when the traffic network is modified or expanded.

2.2 Intersection level

TSTs in this category break the signal optimization problem into sub-problems which are assigned to intersection controllers.

2.2.1 Conventional TSTs with no coordination

Several academic papers have proposed agent-based solutions where isolated smart intersection controllers execute decision making algorithms to benefit their respective intersections [2,10,15,23,24,42]. The proposed approaches have been validated on simple simulated grid networks or single intersections, using simplistic assumptions about traffic. In addition, optimizations at isolated intersections (without any knowledge about other intersection states) do not guarantee an optimization at the network level.

2.2.2 Conventional TSTs with implicit coordination

PRODYN [20]'s optimization at the intersection level uses improved forward dynamic programming with constraints on maximum and minimum greens. The coordination between controllers is implicit. It is performed by (a) simulating a specific intersection output as soon as the optimization is computed, and sending the simulation output to each downstream controller; (b) using the output message from upstream controllers at the next time step to forecast arrivals. Although PRODYN's approach is conceptually applicable to an entire set of intersection controllers, the exponential complexity of dynamic programming limits its applicability to only a few intersections.

OPAC (Optimized Policies for Adaptive Control) [18] was the first comprehensive strategy to be developed in the U.S. for real-time, adaptive TST. OPAC has gone through several development cycles ranging from OPAC I to OPAC-VFC (Virtual Fixed Cycle). OPAC's intersection controller strategy features a dynamic optimization algorithm that calculates signal timings to minimize a performance function for delay and vehicle stops. The controller's algorithm uses measured as well as predicted traffic data. It determines phase durations that are constrained only by minimum and maximum green times. Similarly to PRODYN, OPAC's earlier versions implement implicit coordination. OPAC suffers from the limitations of dynamic programming.

2.2.3 Conventional TSTs with explicit coordination

In OPAC-VFC, the coordination is explicit and is achieved through communication with a central system responsible to identify "critical intersections" and optimizing the cycle length for a group of intersections.

RHODES [25] decomposes the traffic problem into three hierarchical levels. The highest level is the "dynamic network loading" model which captures the slow varying characteristics of traffic (e.g., road closures), and the route selection of travelers. The middle level is the "network flow control" which captures traffic flow characteristics in terms of platoon of vehicles and their speed. The lower level is the "intersection control" which captures fast

varying traffic characteristics in terms of individual vehicles. Each level makes use of prediction models. RHODES and OPAC do not employ defined traffic cycles or signal timing plans. They utilize traffic flow models that predict vehicle arrivals at the intersection, and adjust the timing of each phase to optimize an objective function. Because they emphasize traffic prediction, these systems can respond to the natural statistical variations in traffic flow as well as to flow variations caused by traffic incidents or other unpredictable events. Intersection control equipment for these systems is more complex and not readily available in the field.

2.2.4 AI-based TSTs

With respect to intersection-level TSTs that implement AI-based techniques, most recent approaches are research-oriented and heavily based on the use of the multi-agent system paradigm. The core concept for these systems is that intersection controllers are controlled by autonomous software agents that are capable of interacting with one another to achieve a local or global goal. Models and architectures have been presented [17,32], and solutions proposed [1,6,27,28]. Unfortunately, these solutions are based on assumptions that simplify the traffic signal optimization problem and were validated on simple networks. Other systems such as [19,21] consider real-world traffic constraints but were validated on simple grid-based networks. Only a very few multi-agent solutions have considered the full spectrum of real-world traffic constraints and were deployed or validated on simulated models of real cities. We discuss them below.

In SURTRAC [33,43], the intersection control optimization is formulated as a scheduling problem where each intersection is considered as a single machine, and platoons of vehicles as "non-divisible" jobs. Intersection controllers receive information about incoming vehicles from their direct neighbors and use forward dynamic programming to calculate near optimal schedules. In SURTRAC, the interactions are limited to traffic-data exchange between direct neighbors, and scheduling is done in isolation, at the intersection level. In addition, from a deployment perspective, advanced detectors placed on the upstream end points of entry approaches are needed. The SURTAC approach is extended in [21] to incorporate bi-directional information exchange whereby agents communicate their outflows to the downstream neighbor agents as a prediction of future load, and downstream agents communicate their current congestion level to upstream agents. Communications and decision making only consider direct neighbors and the approach has been validated on a simple two-way grid network including 24-intersections.

With respect of RL-based approaches, the basic premise is that traffic signal timing is not pre-defined, but agents learn the appropriate traffic signal settings. In [13], Dusparic and Cahill discuss DWL, a multi-agent RL-based algorithm for multi-policy optimization. In DWL, agents collaborate to satisfy multiple heterogeneous policies simultaneously (e.g., prioritize buses, reduce pollution). An agent uses a combination of Q-learning and W-learning processes for each of its local policies, and to learn the suitability of its actions for each of its direct neighbor's policies. Collaboration is implicit and restricted to immediate neighbors. It is achieved through the concept of "remote policy". With respect to traffic signal timing, the authors state that the optimization is related to phase selection but no details is provided about the process. The emphasis of this paper is more on the RL-based multi-policy optimization than signal timing optimization. DWL was evaluated on a simulated map of Dublin's inner city including 62 signalized intersections. The policies used in the evaluation are a policy that optimizes global waiting time and a policy that priotitizes public transport vehicles. Experiments ran using artificially generated data show that DWL outperforms the traditional fixed-timed approach and the Simple Adaptive Technique [30].

In [14], the authors extend DWL to consider the optimization of phase duration. This optimization is only possible if more fine-grained traffic data (i.e., precise traffic counts) is available. The extended DWL called REALT was evaluated in VISSIM [29] on a simulated model of Cork City comprising six intersections. The same policies as in [13] were implemented. Experiments ran using real-world data show that REALT outperforms SCOOT in terms of delay and number of stops.

El-Tantawy et al. [16] present a coordinated multi-agent reinforcement learning architecture called MARLIN-ATSC. In MARLIN-ATSC, agents can operate in either independent or integrated mode. Coordination is implicit and achieved through multi-agent modular Q-learning. In modular Q-learning, the state space is partitioned into partial state spaces comprising two agents. An agent learns a joint policy with only one of its direct neighbors. With respect to traffic signal timing, the optimization is related to the selection of a phase among a set of pre-defined phases. MARLIN-ATSC was tested on a simulated network of the Lower Downtown Toronto network comprising 59 intersections. Real-world data for about 25,000 vehicle trips during morning peak hours was used to evaluate the system. Experimental results show that MARLIN-ATSC reduces the average intersection delay compared to a basic signal timing used by the City of Toronto. The main drawback of MARLIN-ATSC is the assumption that an intersection controller can only consider the interest of one immediate neighboring controller.

In addition to the limitations discussed above, both [16] and [14] assume that the systems may have variable phasing sequence. While this assumption may be reasonable in a simulated environment, it is not acceptable in a real-world setting [22]. A variable phasing sequence can lead to endless greens for phases with continuous demand.

In this paper, we present DALI, a coordinated agent-based traffic signal timing model for alleviating urban traffic congestion. Our proposed model is implemented in MATISSE 3.0, a large-scale multi-agent based simulation system. The unique characteristics of DALI are: (1) it is defined based on real-world parameters and constraints of an existing TST, and will be deployed as an extension of that TST; (2) it is fully decentralized and employs explicit controller-to-controller coordinations that span more than the direct neighbors. As such, it considers the feedback of all agents that may be affected by a potential signal timing change; (3) unlike multi-agent RL systems, its does not allow variable phasing sequence. In addition, DALI's signal timing optimization involves more than phase selection; and (4) it has been validated by traffic engineers as well as through extensive agent-based simulation of a real city with the largest road network simulated so far.

3 Agent algorithms

In DALI, agents communicate with each other through direct links and do not have a supervising agent to oversee coordination. Agents have knowledge of the traffic network topology. They receive information about the incoming traffic flow from their neighboring controllers and determine the outgoing traffic flow based on the data sensed by their inductive loops which are positioned a few feet before the stop bar (see Fig. 2). Intersections are assigned weights to indicate their criticality in the traffic network. By default, agents execute a pre-timed, semi-actuated or fully-actuated strategy. The choice of strategy depends on the intersection's attributes and the time of day. At the same time, they observe the status of traffic at their respective intersections. If they detect congestion they execute the DALI strategy [38,39].

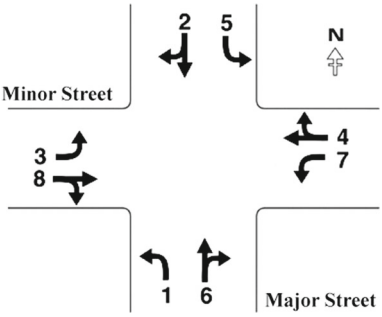

Fig. 1 Standard phases at a four-legged intersection

Before discussing the DALI model and the agent's algorithms, we first give a brief overview of the traffic engineering terminology.

3.1 Concept definitions

The definitions, given in this section are based on the U.S. Department of Transportation Traffic Signal Timing Manual [22] and the City of Richardson's Traffic & Transportation procedures.

Timing parameters

Phase A controller timing unit associated with the control of one or more movements (i.e., through movement, right turn movement) at an intersection. Most controllers sold today provide eight phases to serve standard four-legged intersections (see Fig. 1).
Minimum green The first timed portion of the *green interval* which may be set in consideration of the number of vehicles between the phase detector and the stop line.
Maximum green This time setting defines the maximum length of time that a phase can be green where there is a demand for a conflicting vehicle flow.

Coordination of traffic signal phases

It is the ability to synchronize multiple intersections to enhance the operation of one or more directional movements in a traffic system. In general terms, there are three basic parameters that, when taken together, define a coordinated traffic signal plan. These are:

Cycle length This is the total time to complete one sequence of signalization around an intersection.
*Offset*This is the time relationship, expressed in seconds, between coordinated phases at subsequent traffic signals.
Split This is the time assigned to a phase during coordinated operations.

It is important to note that in the context of traffic management, coordination refers to the setting of the above mentioned parameters. It does not correspond to the coordination concept used in multi-agent systems.

Traffic signal operation modes

Traffic signals operate in various modes.

In *pre-timed mode*, phases and cycles are pre-set according to a predetermined schedule, based on historic traffic patterns.
In *semi-actuated mode*, detectors are placed only on the main street approaches. The main street has green until the actuation of a side street detector. The side street then receives a green phase until either all vehicles are served (gap out) or a preset maximum green is reached (max out).
In *fully actuated mode*, all approaches have detectors. The signal phases are controlled by detector actuations. Minimum greens and maximum greens are specified for each phase.

3.2 Model definition

$T = \{t_1, \ldots, t_i\}$ is the set of time-stamps at which traffic conditions are evaluated.

$C = \{c_1, \ldots, c_n\}$ is the set of intersection controllers. An intersection controller c_n is assigned a weight ω which corresponds to its priority in the road network.

$Rd = \{r_{c_1,c_2}, \ldots, r_{c_m,c_n}\}$ is the set of road segments between intersections.

$LN_{r_{c_m,c_n}} = \{r_{c_m,c_n}.ln_1, \ldots, r_{c_m,c_n}.ln_w\}$ is the set of lanes for road segment r_{c_m,c_n}.

$PH_{c_n} = \{ph_{c_n,1}, \ldots, ph_{c_n,k}\}$ is the set of phases for the intersection controlled by c_n.

A phase $ph_{c_n,k}$ is defined in terms of γ, the split time, ν, the minimum green time, η, the maximum green time, the yellow time, the red time and $LN_{ph_{c_n,k}}$, the set of lanes it applies to.

$p(r_{c_m,c_n}.ln_w, r_{c_n,c_p}.ln_u)$ is the probability that a vehicle exiting lane w in road segment r_{c_m,c_n} enters lane u in road segment r_{c_n,c_p}. This probability is computed by traffic engineers based on historical data.

$p(r_{c_m,c_n}, r_{c_m,c_n}.ln_w)$ is the probability that a vehicle which enters road segment r_{c_m,c_n}, leaves it from lane w. This probability is also computed by traffic engineers based on historical data.

$rateOut(t_i, \tau, r_{c_m,c_n}.ln_w)$ is the rate of vehicles (per second) that can leave the intersection through lane w of road segment r_{c_m,c_n} within the time interval τ that ends at time t_i.

$rateIn(t_i, \tau, r_{c_m,c_n})$ is the rate of vehicles (per second) that enter road segment r_{c_m,c_n} in the time interval τ that ends at time t_i.

$\xi_{r_{c_m,c_n}.ln_w}(t_i, \tau)$ is the *traffic flow rate* for lane $r_{c_m,c_n}.ln_w$, i.e., the ratio of vehicles getting in and leaving the lane. It is defined as:

$$\xi_{r_{c_m,c_n}.ln_w}(t_i, \tau) = \frac{rateIn(t_i, \tau, r_{c_m,c_n}) \times p(r_{c_m,c_n}, r_{c_m,c_n}.ln_w)}{rateOut(t_i, \tau, r_{c_m,c_n}.ln_w)}$$

3.3 DALI agent algorithms

In DALI, agents collaborate with one another to dynamically respond to traffic changes. In this section we discuss the agent algorithms at the basis of the collaborative approach. The

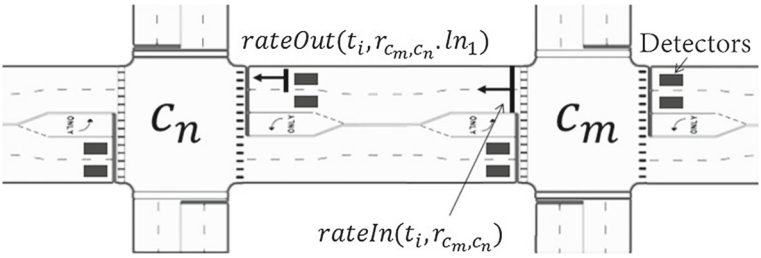

Fig. 2 c_n determines $rateOut$ and receives $rateIn$ from c_m

algorithms make use of parameters (i.e., a, b, h, d, e, f and g) which are defined by traffic engineers based on historical traffic data.

At any given time, if an agent determines that its intersection is congested, it deliberates and defines a timing plan to alleviate congestion by adjusting splits. Then, it broadcasts the plan to the neighboring agents. Upon receipt, the agents evaluate the plan by calculating its effect on each of their intersections outgoing roads. They in turn communicate the information with the affected neighboring agents. And the process iterates until it either reaches (a) an intersection within the city boundaries for which the effect of the request is below a threshold or (b) an exit junction at the citys boundaries. The information is then propagated back, and at each stage of the propagation, the agents consider each others feedback for their decision on their *level of agreement* with the plan, i.e., a value which indicates the extent to which an agent can agree with the terms of the plan. The initiating agent then decides whether to execute or ignore the plan. It proceeds by informing the agents of its final decision and, in case the plan is to be applied, requests that they update their timing accordingly.

Algorithm 1 Controller Congestion Reduction

Require: PH_{c_n}, t_i
for all $ph_{c_n,k} \in PH_{c_n}$ **do**
  $EvaluateTraffic(ph_{c_n,k}, t_i : TotalInstCong)$
  if $\frac{TotalInstCong}{b} > d$ **then**
    $GeneratePlan(ph_{c_n,k}, t_i : plan_{new})$
    $RequestForEvaluation(ph_{c_n,k}, plan_{new} : \Psi_{c_n})$
    if $\Psi_{c_n} > h$ **then**
      $ExecutePlan(plan_{new})$
    end if
  end if
end for
if $ReceiveRequestForEvaluation(c_p, \kappa_{r_{c_p,c_n}}, \kappa_{ph_{c_q,j}})$ **then**
  $ComputeLevelOfAgreement(\kappa_{r_{c_p,c_n}}, \kappa_{ph_{c_q,j}})$
end if
if $ReceiveRequestForExecution(c_p, plan_{new})$ **then**
  $AdjustTiming(plan_{new})$
end if

3.3.1 Detecting Congestion

Intersection controller c_n continuously evaluates the traffic state by executing Algorithm 1 to determine if a re-timing operation is necessary. As shown in Fig. 2, at each t_i, c_n receives $rateIn$ (determined by its neighbors' detectors) and determines $rateOut$.

Algorithm 2 Evaluate Traffic

Require: PH_{c_n}, t_i
1: **for all** $ph_{c_n,k} \in PH_{c_n}$ **do**
2: $TotalInstCong \leftarrow 0$
3: **for** $j = 0$ *to* b **do**
4: $\delta = 0$
5: **for each** $r_{cm,cn}.ln_w \in LN_{ph_{cn,k}}$ **do**
6: $\delta \leftarrow \xi_{ti-j,r_{cm,cn}.ln_w} + \delta$
7: **end for**
8: $Cong_{t_i,ph_{c_n,k}} \leftarrow \delta$
9: **if** $Cong_{t_i,ph_{c_n,k}} \geq a$ **then**
10: $TotalInstCong \leftarrow TotalInstCong + 1$
11: * TotalInstCong Represent Sum Over InstantCongestion
12: **end if**
13: **end for**
14: **end for**

At time t_i, controller c_n computes $Cong_{t_i,ph_{c_n,k}}$ as the sum of throughputs for the set of lanes controlled by $ph_{c_n,k}$.

$$Cong_{t_i,ph_{c_n,k}} = \sum_{r_{cm,cn}.ln_w \in LN_{ph_{c_n,k}}} \xi_{t_i,r_{cm,cn}.ln_w}$$

If $Cong_{t_i,ph_{c_n,k}}$ is greater than threshold a, then c_n considers that there is an *instant congestion* and assigns the value of 1 to $InstantCongestion$ defined as:

$$InstantCongestion_{t_i,ph_{c_n,k}} = \begin{cases} 1 & Cong_{t_i,ph_{c_n,k}} \geq a \\ 0 & Cong_{t_i,ph_{c_n,k}} < a \end{cases}$$

It proceeds by considering the past b evaluation cycles to determine the percentage of evaluation cycles in which the phase was congested (see Fig. 3). This is defined as:

$$PercentCong_{t_i,ph_{c_n,k}} = \frac{\sum_{j=i-b}^{i} InstantCongestion_{t_j,ph_{c_n,k}}}{b} \times 100$$

If $PercentCong_{t_i,ph_{c_n,k}} > d$ then the road lanes controlled by $ph_{c_n,k}$ are considered to be congested.

To illustrate the various steps, we use a section of the City of Richardson's road network (See Fig. 4). As shown in Fig. 5, c_2 has four incoming roads. The four phases for c_2's intersection are $\{ph_{c_2,1}, ph_{c_2,2}, ph_{c_2,3}, ph_{c_2,4}\}$. These phases apply as follows: $ph_{c_2,1}$ for r_{c_6,c_2}, $ph_{c_2,2}$ for r_{c_3,c_2}, $ph_{c_2,3}$ for r_{c_5,c_2} and $ph_{c_2,4}$ for r_{c_1,c_2}. The phases have the following attribute values: the split $\gamma = 40$, the minimum green $\nu = 20$, the maximum green $\eta = 60$. Thresholds a, b and d have the values of $a = 0.6$, $b = 100$ and $d = 80$.

In this example, c_2 evaluates the status of its intersection at the time-stamp t_{4100} and within the time interval $\tau = 500$. It starts with phase, $ph_{c_2,1}$ and calculates the average traffic

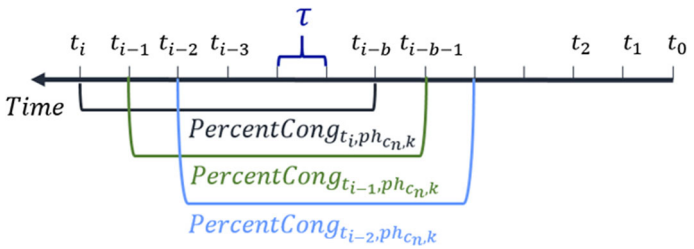

Fig. 3 $PercentCong$ of phase $ph_{c_n,k}$

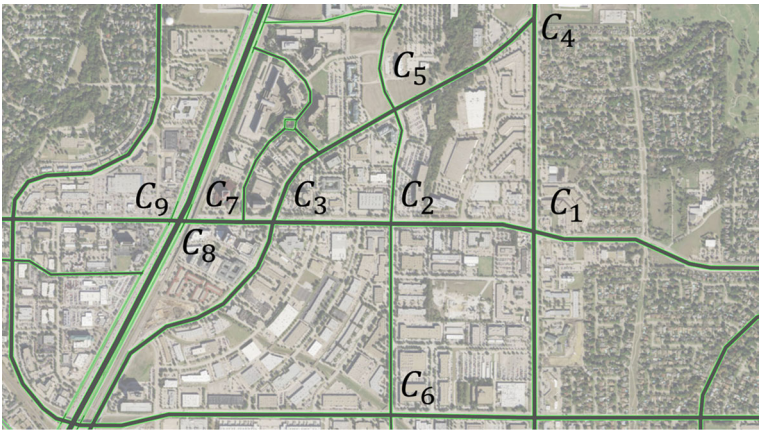

Fig. 4 Overview of the network in the case study

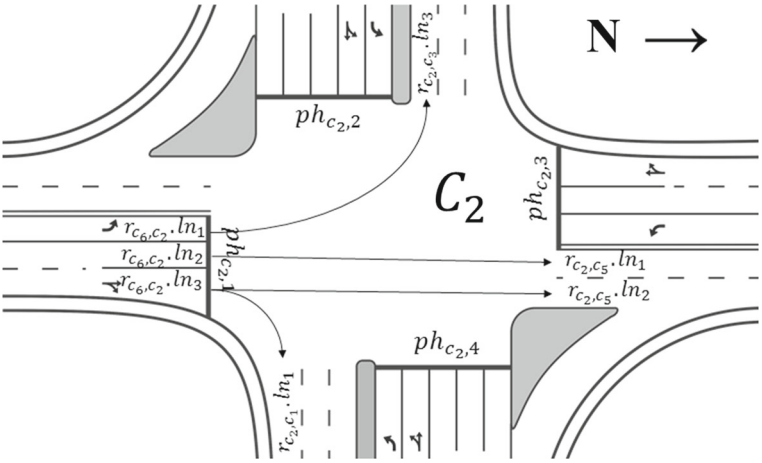

Fig. 5 Intersection assigned to c_2

throughput $Cong_{t_{4100},ph_{c_2,1}}$ for the set of road lanes that $ph_{c_2,1}$ controls. Given that $rateOut(t_{4100}, 500, r_{c_6,c_2}.ln_3) = 0.8$, $p(r_{c_6,c_2}, r_{c_6,c_2}.ln_3) = 0.2$ and $rateIn(t_{4100}, 500, r_{c_6,c_2}) = 2.4$, the value of $\xi_{t_{4100},500,r_{c_6,c_2}.ln_3}$ is:

$$\xi_{t_{4100},500,r_{c_6,c_2}.ln_3} = \frac{2.4 \times 0.2}{1} = 0.48$$

For the sake of illustration, we assume that $Cong_{t_{4100},ph_{c_2,1}} = 0.83$ which is greater than threshold $a = 0.6$. Controller c_2, then retrieves the calculated values of $Cong$ between time stamps t_{4100} and t_{4000} and finds that 91 of them are greater than a. Therefore,

$$PercentCong_{t_{4100},ph_{c_2,1}} = \frac{91}{100} \times 100 > 80$$

Consequently, c_2 detects congestion on phase $ph_{c_2,1}$ and deliberates to define a new plan.

3.3.2 Generate a new plan

The controller deliberates to determine the value of a new split that will alleviate congestion on $ph_{c_n,k}$. This is achieved in step 7 of Algorithm 3. The value of the new split is calculated as:

$$plan_{new}.phase.\gamma = plan_{cur}.phase.\gamma \times \left(e + \frac{\sum_{j=i-\nu}^{i} Cong_{t_j,ph_{c_n,k}}}{\nu} \times f\right)$$

In order to determine the value of a new split, c_n starts by determining the severity of its congested status by computing the average congestion level over the last ν evaluation cycles ($\frac{\sum_{j=i-\nu}^{i} Cong_{t_j,ph_{c_n,k}}}{\nu}$) and increases its current split time ($plan_{cur}.phase.\gamma$) proportionally to this value. The more congested a road segment has been, the more additional green time is needed to release vehicles and alleviate congestion. e and f coefficients defined by traffic engineers based on historical data. They regulate the influence of the traffic throughput and the current split time for the new split time. Values of cycle length and offset change in the new split. If $plan_{new}.phase.\gamma$ is greater than the maximum allowed split time γ_{MAX} defined for phase $ph_{c_n,k}$ as:

$$ph_{c_n,k}.\gamma_{MAX} = ph_{c_n,k}.\eta + ph_{c_n,k}.\epsilon + ph_{c_n,k}.\xi$$

then its value is set to $ph_{c_n,k}.\gamma_{MAX}$ (step 9).

Algorithm 3 Generate Plan

Require: $ph_{c_n,k}, t_i$
Ensure: $plan_{new}$
1: $plan_{new}.phase \leftarrow ph_{c_n,k}$
2: $\chi \leftarrow 0$
3: **for** $j = i - \nu$ *to* i **do**
4: $\quad \chi \leftarrow \chi + Cong_{t_j,ph_{c_n,k}}$
5: **end for**
6: $\chi \leftarrow \frac{\chi}{\nu}$
7: $plan_{new}.phase.\gamma \leftarrow plan_{cur}.phase.\gamma * (e + \chi * f)$
8: **if** $plan_{new}.phase.\gamma > ph_{c_n,k}.\gamma_{MAX}$ **then**
9: $\quad plan_{new}.phase.\gamma \leftarrow ph_{c_n,k}.\gamma_{MAX}$
10: **end if**

In the example above, we assume that the average $Cong$ for phase $ph_{c_2,1}$ in the last $\nu = 10$ evaluation cycles is 0.9. Given that $e = 1$ and $f = 0.33$, c_2 defines a new plan for $ph_{c_2,1}$, and computes $plan_{new}.phase.\gamma$ as:

$$plan_{new}.phase.\gamma = 40 + (1 + 0.9 \times 0.33) \approx 52$$

Therefore, c_2 determines that it needs to increase $ph_{c_2,1}.\gamma$ by 12 seconds.

3.3.3 Request for evaluation

c_n determines the impact of executing the new plan on the neighboring intersections in terms of κ, the increment in vehicle rate. This increment is computed as the rate of vehicles that leave the intersection through lane $r_{c_m,c_n}.ln_w$ times the additional green time needed to alleviate the congestion computed as $plan_{new}.phase.\gamma - plan_{cur}.phase.\gamma$ divided by the new green time. $\kappa_{r_{c_m,c_n}.ln_w}$ is calculated for road lane $r_{c_m,c_n}.ln_w$ as:

$$\kappa_{r_{c_m,c_n}.ln_w} = \frac{rateOut(t_i, r_{c_m,c_n}.ln_w)}{plan_{new}.phase.\gamma} \times (plan_{new}.phase.\gamma - plan_{cur}.phase.\gamma)$$

$\kappa_{ph_{c_n,k}}$ for a phase $ph_{c_n,k}$ is defined as the sum of $\kappa_{r_{c_m,c_n}.ln_w}$ for the set of lanes controlled by the phase (Algorithm 4, Step 3). In the same way, $\kappa_{r_{c_n,c_p}}$ for a road segment r_{c_n,c_p}, is the sum of $\kappa_{r_{c_n,c_p}.ln_w}$ (Algorithm 4, Step 10).

Controller c_n proceeds by sending $plan_{new}$, $\kappa_{r_{c_n,c_p}}$ and $\kappa_{ph_{c_n,k}}$ to each adjacent controller c_p for evaluation. $\kappa_{ph_{c_n,k}}$ corresponds to the increment in the rate of vehicles that exits the road lanes controlled by $ph_{c_n,k}$, in case the new plan is to be executed. $\kappa_{r_{c_n,c_p}}$ corresponds to the portion of $\kappa_{ph_{c_n,k}}$ that goes to road segment r_{c_n,c_p}.

Algorithm 4 Request for Evaluation

Require: $ph_{c_n,k}$, $plan_{new}$
Ensure: Ψ_{c_n}
1: $\kappa_{ph_{c_n,k}} \leftarrow 0$
2: **for each** $r_{c_m,c_n}.ln_w$ *in* $LN_{ph_{c_n,k}}$ **do**
3: $\kappa_{ph_{c_n,k}} \leftarrow \kappa_{ph_{c_n,k}} + \kappa_{r_{c_m,c_n}.ln_w}$
4: **end for**
5: $\Psi_{c_n} \leftarrow 0$
6: **for each** *accessible neighbor* c_p , *in parallel* **do**
7: $\kappa_{r_{c_n,c_p}} \leftarrow 0$
8: **for** $r_{c_m,c_n}.ln_w \in LN_{ph_{c_n,k}}$ **do**
9: **for** $r_{c_n,c_p}.ln_u \in LT_{r_{c_m,c_n}.ln_w}$ **do**
10: $\kappa_{r_{c_n,c_p}} \leftarrow \kappa_{r_{c_n,c_p}} + (p(r_{c_m,c_n}.ln_w, r_{c_n,c_p}.ln_u) \times \kappa_{r_{c_m,c_n}.ln_w})$
11: **end for**
12: **end for**
13: **Send**$(c_p, \kappa_{r_{c_n,c_p}}, \kappa_{ph_{c_n,k}})$
14: **Receive**(c_p, Ψ_{c_p})
15: $\Psi_{c_n} \leftarrow \Psi_{c_n} + \Psi_{c_p}$
16: **end for**

In the illustrative example, c_2 proceeds by calculating $\kappa_{r_{c_1,c_2}.ln_1}$ as:

$$\kappa_{r_{c_1,c_2}.ln_3} = \frac{0.8 \times (52 - 40)}{52} = 0.18$$

Given that $\kappa_{r_{c_6,c_2}.ln_2} = 0.32$ and $\kappa_{r_{c_6,c_2}.ln_1} = 0.16$, $\kappa_{ph_{c_2}.1}$ takes the value of 0.66. Controller c_2 then calculates the effect of executing a new plan on its neighboring intersections, including c_5. Assuming $p(r_{c_6,c_2}.ln_3, r_{c_2,c_5}.ln_2) = 0.5$, $p(r_{c_6,c_2}.ln_3, r_{c_2,c_1}.ln_1) = 0.5$, $p(r_{c_6,c_2}.ln_2, r_{c_2,c_5}.ln_1) = 1$ and $p(r_{c_6,c_2}.ln_2, r_{c_2,c_1}.ln_1) = 0$, $\kappa_{r_{c_2,c_5}}$ is calculated as:

$$\kappa_{r_{c_2,c_5}} = 0.5 \times 0.18 + 1 \times 0.32 = 0.41$$

c_2 then sends a request for evaluation to c_5 with $\kappa_{r_{c_2,c_5}} = 0.41$ and $\kappa_{ph_{c_2}.1} = 0.66$. This indicates that, by executing $plan_{new}$, an additional 0.66 vehicle per seconds (vps) will leave the road controlled by $ph_{c_2,1}$, and out of the 0.66 vps, 0.41 vps will enter r_{c_2,c_5}.

3.3.4 Compute level of agreement

Upon receipt of a new plan, c_n's neighboring controller c_p computes $\kappa_{r_{c_p,c_q}}$ for each of its neighbor controllers c_q and request that they each evaluate the plan. The process propagates until at a given intersection, either the value of κ is smaller than threshold g or the plan reaches the road network boundaries. Following this step and recursively, each controller sends back its level of agreement in terms of a real number Ψ, to the controller from which it has received the request. An intermediate controller, c_p, calculates Ψ_{c_p} based on the existing traffic throughput, its priority ω and the ratio of the received additional vehicle throughput.

A controller agent, c_p, calculates the agreement level Ψ_{c_p} based on the existing traffic throughput, its priority ω and the ratio of the received additional vehicle throughput. The core computation of the agreement level is c_ps capacity to handle an extra load of vehicles which is computed as ratio of the rate of incoming vehicles ($rateIn(t_i, \tau, r_{c_n,c_p})$) plus the expected additional vehicles ($\kappa_{r_{c_n,c_p}}$), divided by the rate of outgoing vehicles ($rateOut(t_i, r_{c_n,c_p}.ln_u)$)). This ratio is adjusted to reflect decreasing values if the total rate of incoming vehicles is much higher than the rate of outgoing vehicles. It is further adjusted to account for the criticality $\omega(c_p)$ of the intersection, i.e., the more critical the intersection and therefore the higher the value of $\omega(c_p)$, the more emphasized is the capacity value. The same applies to $\frac{\kappa_{r_{c_n,c_p}}}{\kappa_{ph_{c_n,k}}}$ which corresponds to the portion of the total additional load that is expected to be received, i.e., the higher $\frac{\kappa_{r_{c_n,c_p}}}{\kappa_{ph_{c_n,k}}}$ the more emphasized is the capacity value.

x, y and z are coefficients that calibrate the influence of variables in Ψ. After receiving the level of agreement from all affected neighbors, c_p adds them to its own level of agreement Ψ_{c_p} and sends the value back to c_n. The final decision is made based on the value of Ψ_{c_n} representing the opinion of all affected controllers in the network.

In the illustrative example, c_5 receives the request for the new timing plan. It calculates Ψ_{c_5} using the current $rateOut(t_{4100}, 500, r_{c_2,c_5}.ln_1) = 1, rateOut(t_{4100},,500, r_{c_2,c_5}.ln_2) = 0.3$, $rateIn(t_{4100}, 500, r_{c_2,c_5}) = 1.2$, $p(r_{c_2,c_5}, r_{c_2,c_5}.ln_1) = 0.8$ and $p(r_{c_2,c_5}, r_{c_2,c_5}.ln_2) = 0.2$. Ψ_{c_5} is calculated as:

$$\begin{aligned}\Psi_{c_4} &= 1.0 \times 2.0 \times \frac{0.41}{0.66} \times \left(\left(1 - 1 \times \frac{(0.41 + 1.2) \times 0.8}{1}\right)\right. \\ &\quad \left. + \left(1 - 1 \times \frac{(0.41 + 1.2) \times 0.2}{0.3}\right)\right) \\ &= -1.26\end{aligned}$$

c_5 proceeds by calculating κ for c_3, c_4. If κ is greater than threshold g, c_5 requests that they evaluate the plan. c_3 and c_4's responses are added to Ψ_{c_5} and sent back to c_2. Upon

Algorithm 5 Compute Level Of Agreement

Require: $\kappa_{r_{c_n,c_p}}, \kappa_{ph_{c_n,k}}$
Ensure: Ψ_{c_p}
$\Psi_{c_p} \leftarrow 0$
for $r_{c_n,c_p}.ln_u \in LN_{r_{c_n,c_p}}$ **do**
  $\Psi_{c_p} \leftarrow \Psi_{c_p} + x \times \omega(c_p) \times \frac{\kappa_{r_{c_n,c_p}}}{\kappa_{ph_{c_n,k}}} \times (y - z \times \frac{(\kappa_{r_{c_n,c_p}} + rateIn(t_i, r_{c_n,c_p}) \times p(r_{c_n,c_p}, r_{c_n,c_p}.ln_u)}{rateOut(t_i, r_{c_n,c_p}.ln_u)})$
end for
for each *accessible neighbor* c_q *from* c_p *, in parallel* **do**
  $\kappa_{r_{c_p,c_q}} \leftarrow 0$
  for $r_{c_n,c_p}.ln_u \in LN_{r_{c_n,c_p}}$ **do**
    for $r_{c_p,c_q}.ln_f \in LF_{r_{c_n,c_p}.ln_u}$ **do**
      $\kappa_{r_{c_n,c_p}} \leftarrow \kappa_{r_{c_n,c_p}} + p(r_{c_n,c_p}, r_{c_n,c_p}.ln_u) \times p(r_{c_n,c_p}.ln_u, r_{c_p,c_q}.ln_f) \times \kappa_{r_{c_n,c_p}}$
    end for
  end for
  if $\kappa_{r_{c_n,c_p}} > g$ **then**
    Send$(c_q, \kappa_{r_{c_n,c_p}}, \kappa_{ph_{c_n,k}})$
    Receive(c_q, Ψ_{c_q})
    $\Psi_{c_p} \leftarrow \Psi_{c_p} + \Psi_{c_q}$
  end if
end for
Send(c_n, Ψ_{c_p})

receipt of Ψ_{c_5}, Ψ_{c_1} and Ψ_{c_3}, controller c_2 calculates Ψ_2. Negative values of Ψ are considered as a level of disagreement. Having $\Psi_2 = 2.34$, c_2 executes the new plan and announces the execution to all controllers in the network which in turn adapt their timing plans.

3.4 Special cases

During the execution of the scenario discussed above, several exceptions may occur. These include the following [35]:

1. A controller may receive more than one plan to evaluate at the same time. In this case, the controller evaluates the plan sent by the controller with the highest priority and halts the evaluation of other plans. If the plans were sent by neighbor controllers having the same level of criticality, the controller selects one according to a pre-defined criteria, e.g., the smaller controller ID.
2. A controller may receive more than one request to evaluate the same plan. In the example mentioned above, executing c_2's plan will increase the throughput of both r_{c_2,c_3} and r_{c_5,c_3}. This will result in c_3 receiving an evaluation request first from c_2 and then from c_5. Controller c_3 evaluates the request from c_2 and stores the received additional throughput value. Then c_3 considers the stored value to evaluate the request of c_5.
3. A controller may lose connection from the network and stop responding to requests of evaluation. In this case, other agents assume that the disconnected agent fully disagrees with any retiming plan.
4. When a plan gets rejected, the main agent generates a new plan by reducing the split of the rejected plan and ask other agents to evaluate the new plan.
5. After executing a new timing plan, when the traffic situation goes back to normal, the main agent switch back to the original timing plan and ask other agents to do the same.

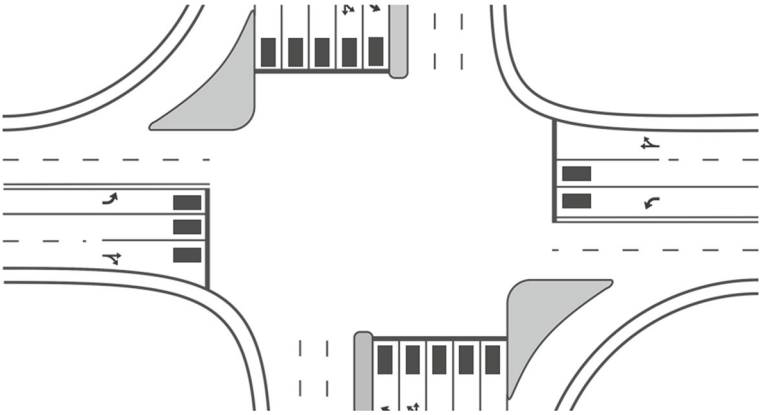

Fig. 6 Induction loops and vehicle sensing area

4 Evaluation

4.1 The city of Richardson's traffic signal timing system

The City of Richardson is located 15 miles north of downtown Dallas and is part of the Dallas-Fort Worth Metroplex. The city has four major highways, eleven major and 6 minor arterial roads and 128 intersections with traffic signals.

The 128 SCATS-based intersection control computers (i.e., traffic controllers) are mounted in cabinets at intersections. They run Linux on an ATC-compliant motherboard offering speed, performance and multi-thread capabilities. A central traffic management center communicates with the traffic controllers via a WiMAX wireless network operating in the licensed 4.9 GHz public safety band with about 2.5 GB/s total throughput. Controller-to-Controller communication links exist but are not used in the current traffic system. Traffic controllers operate in various modes. During the day, a variety of pre-timed plans designed to address variable traffic patterns are executed based on traffic conditions. Past midnight, controllers operate either in pre-timed, semi-actuated or fully-actuated modes depending on the road types and the existence of a detection system.

Vehicles at an intersection are detected through inductive loops. An inductive loop is a coiled wire that is formed into a loop and installed under the surface of roadways at appropriate distances before the stop bar based on the traffic and roadway conditions (see Fig. 6). When a vehicle passes over the loop or is stopped within its area, a pulse is sent to the traffic signal controller signifying the passage or presence of a vehicle. The controller stores the detection information and the time of its occurrence in a local database.

The City of Richardson maintains a traffic count program which conducts scheduled counts on major arterial roads as well as collector streets, i.e., roads which move traffic from local streets to arterial roads. The traffic counts are used for a variety of purposes including the definition of coordinated traffic signal timing along arterial streets.

In order to define traffic signal timing plans, traffic engineers assign values to cycle length, offset and splits based on historical data. Given that inductive loops are positioned a few feet from the stop bar, the vehicles that can be realistically detected are those that cross the inductive loop area. With the inductive loop technology, a complete vehicle count on a road

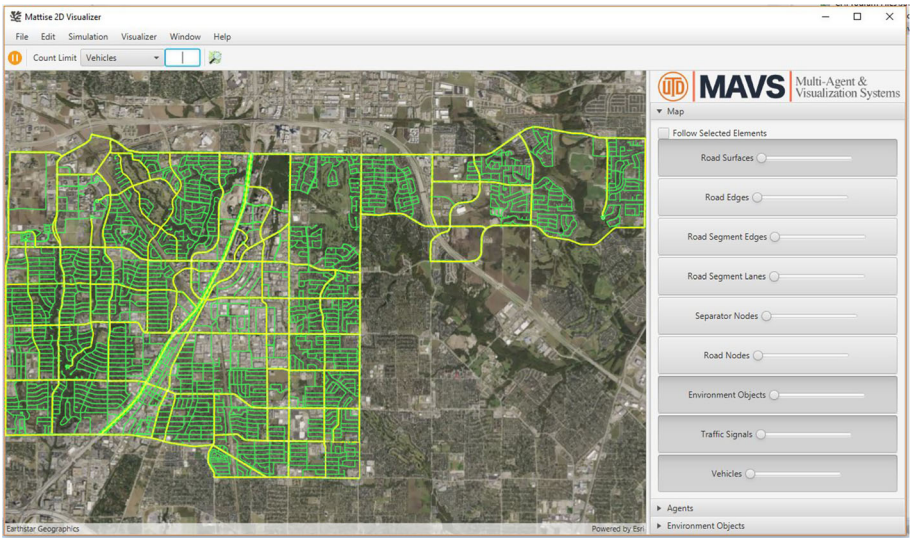

Fig. 7 2D visualization of Richardson's traffic network

Table 1 Number of signalized intersection with various incoming and outgoing lanes

Type	1×1	1×2	1×3	2×2	2×3	3×3
Count	0	4	8	18	29	69

segment is not possible. In addition, except for the induction loop area, the vehicle positions on road segments cannot be obtained.

The measurements that are commonly used in Texas to evaluate the effectiveness of a signal timing plan include *delay*, *queue length* and *number of stops*. In the following sections, we discuss the evaluation of DALI with respect to delay. Similar results were obtained for queue length [35]. Delay is defined as the increment in a vehicle's travel time caused by traffic control devices, compared with the travel time if the vehicle was to maintain its expected speed in the absence of any control device [4].

4.2 Simulation setting

The experiments were run on a multicore PC (Intel Core i7 X980 CPU (3.33 GHz), 6.00 GB, 64-bit Windows 7). A simulated model of the City of Richardson's road network was created in MATISSE. The model includes 1365 road segments and the city's 128 signalized intersections in addition to the 965 non-signalized intersections. Figure 7 shows a 2-D representation of the traffic network. Tables 1 and 2 summarize the types of signalized and non signalized intersections, classified based on the number of incoming and outgoing lanes.

Three simulation settings were run eight times for 86,400 simulation cycles representing a 24-hour time period. The average delay for all vehicles was measured. In the first and second experiment, we use real-world data provided by the City of Richardson to simulate regular traffic patterns with and without accidents. In the third and fourth experiment we simulate continuous random traffic patterns with and without accidents. For all experiments, we com-

Table 2 Number of non-signalized intersection with various incoming and outgoing lanes

Type	1 × 1	1 × 2	1 × 3	2 × 2	2 × 3	3 × 3
Count	533	241	175	16	0	0

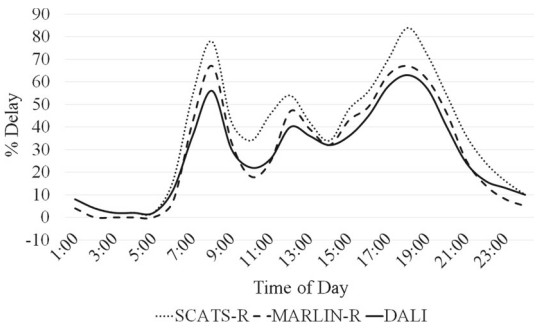

Fig. 8 Average delay using traffic data from the City of Richardson

pare the efficiency of DALI with the SCATS-based system currently in use in Richardson (SCATS-R), and a model of the RL-based MARLIN-ATSC [16] (MARLIN-R). To decrease the learning time of MARLIN agents, we initialized the Q-values based on estimations derived from historical data provided by the City of Richardson.

Experiment 1: Normal traffic conditions
In this experiment, we make use of the traffic data provided by the City of Richardson to determine the number of vehicles in the traffic network at any given time, as well as their distribution in the network. This experiment is intended to analyze the behavior of the three systems under nominal traffic conditions.

As shown in Fig. 8, between the times of 00:30 am and 5:30 am DALI and SCATS-R perform at the same level with respect to delay. This is due to the fact that during this time period, traffic is very light and therefore DALI agents do not perform any action. MARLIN-R agents perform better (53% delay reduction) in this situation because of their flexibility in changing the traffic phases at any time. As we progress during the day (i.e., 6:30 am to 8:30 am) the traffic flow increases, and congestion is detected. DALI agents naturally collaborate with one another to define and implement timing plans that meet the network conditions. As such, DALI performs significantly better than SCATS-R (23% delay reduction). MARLIN-R performs slightly less than DALI. The simulation shows that this is due to the fact that MARLIN-R agents do not handle heavy traffic in small network areas with a large number of intersections efficiently. In those cases, MARLIN-R agents give the right-of-way to vehicles without taking into account the downstream roads which are congested.

Experiment 2: Normal traffic conditions with accident
Figure 9 shows the performance of the systems when an accident is triggered at run time, during normal morning peak traffic. As expected, DALI handles the traffic much better than SCATS-R (35% delay reduction). MARLIN-R agents are unable to control the congestion created by the accident since they have no prior knowledge of the unexpected traffic pattern. Similarly to Experiment 1, the simulation shows that, rather than leading the vehicles towards roads with lighter traffic, MARLIN-R agents send vehicles to congested areas (Fig. 9).

Experiment 3: Continuous random traffic conditions

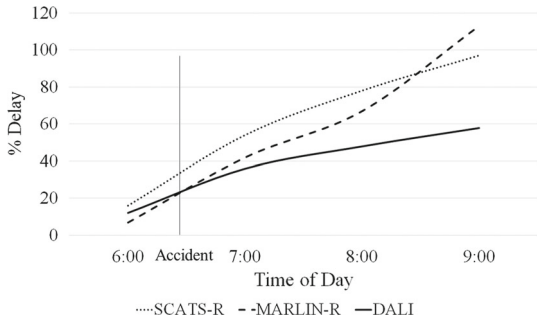

Fig. 9 Average delay with accident in peak morning hours using real traffic data

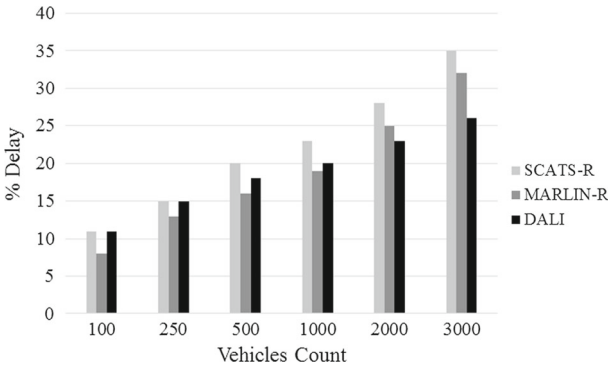

Fig. 10 Average delay for random traffic patterns

In this experiment, the number of vehicles during the simulation remains constant but new vehicles are added randomly while others randomly exit the traffic network. This experiment is intended to illustrate random traffic patterns that are unprecedented. The experiment was run with 100, 250, 500, 1000, 2000 and 3000 vehicles.

Figure 10 shows that when the traffic is light, MARLIN-R agents perform (37%) better because they use a variable phasing sequence. They can extend the current phase or switch to any other phase according to the changes in traffic. On the other hand, SCATS-R controllers and DALI agents execute a fixed phase sequence. Therefore, all phases are executed even in cases where it is not necessary. DALI and SCATS-R perform at the same level in lighter traffic conditions because the controller agents do not detect congestion and therefore, do not change the split. As the number of vehicles increases, DALI agents start to detect congestion and collaborate with other agents for retiming. The collaborative retiming procedure allows DALI to perform better than SCATS. As the number of vehicles increases, MARLIN-R still perform better than SCATS-R. However, DALI do better. This is due to the fact that MARLIN-R agents fail to handle heavy traffic in small, condensed network areas.

Experiment 4: Continuous random traffic conditions with accident Figure 11 shows the performance of DALI, SCATS-R and MARLIN-R in the extreme situation where an accident is randomly triggered in unpredictable traffic conditions. When the traffic is light, the three systems nearly act the same. As traffic gets heavier, DALI operates better than the other two (20% decrease in delay compared to SCATS-R and 12% decrease in delay compared to MARLIN-R). When the number of vehicles reaches 3000, MARLIN-R operates worse that SCATS-R (8% delay increase) because SCATS-R controllers are committed to giving green

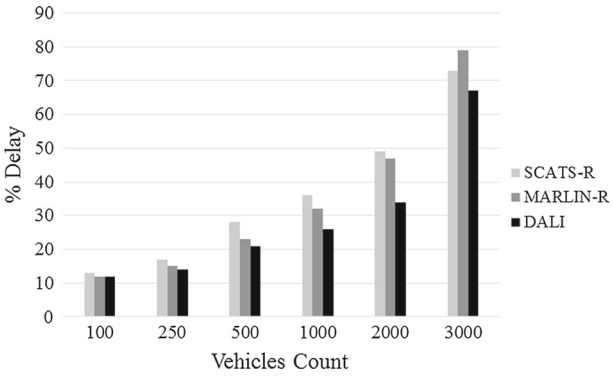

Fig. 11 Average delay for random traffic patterns with accidents

signal to all movements in a cycle whereas MARLIN-R agents lack experience in dealing with new traffic conditions.

4.3 Hybrid simulation

MATISSE is able to run hybrid simulations by retrieving real-time data from deployed controllers. This data which includes the detector states (i.e., active, inactive) and the traffic light state (i.e., green, yellow, red) is processed as follows: when a detector state goes from active to inactive, MATISSE adds a vehicle in the simulation at the detector's position. It also visualizes the queue length at the traffic light.

In the hybrid simulation, simulated vehicles enter the simulation through entry points (represented as red arrows in Fig. 12), and leave the simulation when they reach the exit points (represented by green arrows in Fig. 12). The destinations of the simulated vehicles at the entry points are estimated based on the traffic flow information at the exit points. The performance of DALI in the simulated environment is evaluated by comparing the rate of simulated vehicles that exit the simulated traffic network versus the real world.

4.3.1 Experiment

In this experiment, we create a simulated model of the Waterview corridor in Richardson, which includes three intersections, Frankford Rd, Synergy Pkway and Franklin Jenifer (see Fig. 12). We connect MATISSE to the three real-world controllers, run the hybrid simulation using DALI and compare the results with the actual SCATS-R-based values provided by the controllers.

We ran the hybrid simulation for one week and compared the average queue length and delay in the simulation with their actual SCATS-R counterparts. Figure 13 shows the average traffic flow at different times of the days for network entrance points. As expected, Waterview Parkway gets a rush in the morning from 7:00 A.M. to 9:00 A.M. and in the evening from 4:00 P.M. to 6:00 P.M. Waterview approaches have approximately the same traffic flow during the day; however, traffic drops at nights.

Table 3 shows the reduction in delay for different traffic flows. Similarly to the previous experimental results, when the traffic is light, the average delay does not change since DALI agents do not perform any action. When the traffic flow increases, agents adapt by generating

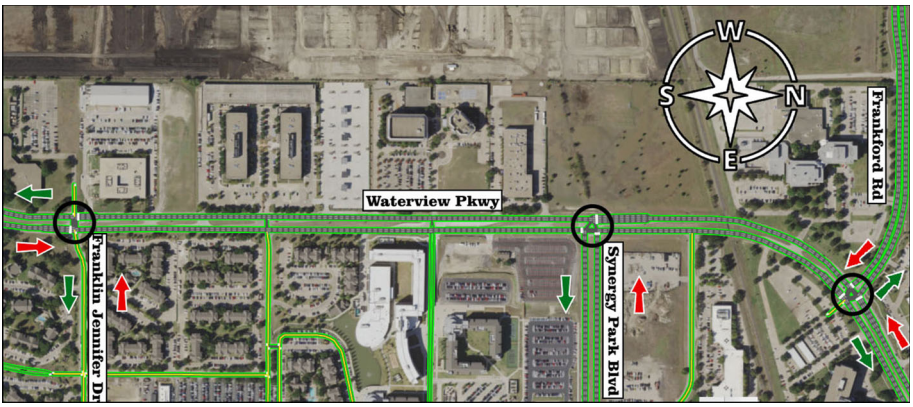

Fig. 12 Arterial image of simulated intersections (Color figure online)

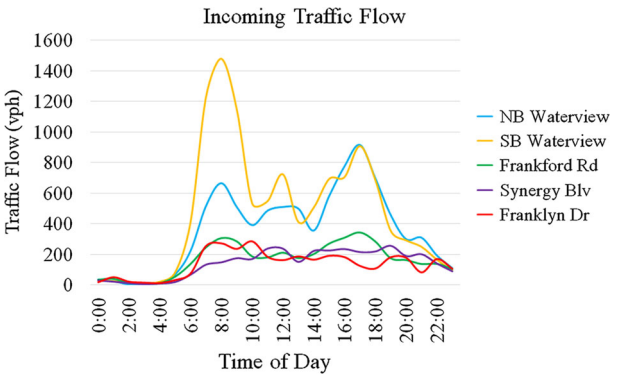

Fig. 13 Average incoming traffic flow at different times of work days

Table 3 Reduction in delay for different traffic flows

Traffic flow (vph)	0–200	200–400	400–600	600–800
% Reduction in delay	0	0	1.27	4.12
Traffic flow (vph)	800–1000	1000–1200	1200–1400	1400–1600
% Reduction in delay	7.15	10.79	17.96	20.81

new plans and executing them. Therefore, the traffic on roads with higher demand get more green which results in a decrease of the average delay.

Figure 14 shows the reduction in queue length for different flow rates at different entry points. As illustrated, the queue length was drastically reduced on both directions at Waterview. However, at the same time, the queue length increased on approaches. The reason is that whenever the traffic flow increases on Waterview, agents react by increasing the green time of the phases that control Waterview. Therefore, the vehicles in the other directions receive less green time.

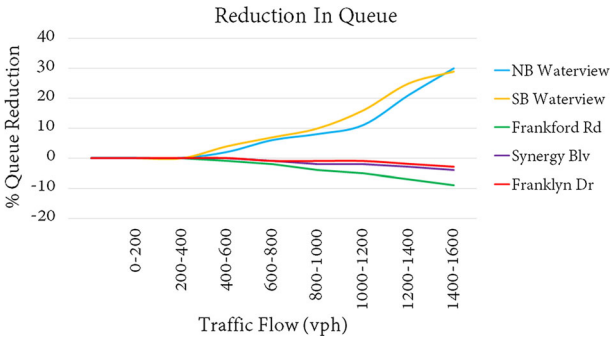

Fig. 14 Delay reduction for different traffic flows at different entrances

5 Conclusion

In this paper we presented DALI, a distributed collaborative multi-agent traffic signal timing (TST) system for highly dynamic traffic conditions. DALI has been validated on a simulated model of City of Richardson's traffic network. The experimental results show that the collaborative multi-agent controllers outperforms the traditional SCATS-based system currently used by the City of Richardson. While a simulated model of MARLIN performs better than DALI in stable traffic conditions with light to normal traffic, the RL-based model does not operate efficiently in two settings: (1) random traffic conditions and (2) nominal traffic conditions with heavy traffic in condensed traffic network areas. Our goal is to investigate the development of a hybrid model that will integrate some RL in the DALI agents.

This work is a first step towards the implementation of an agent-based TST for the City of Richardson. Before the deployment of the first prototype, agent-to-agent communication costs need to be assessed. Our assumption is that, given that the currently deployed SCATS controllers communicate through a WiMAX network with a speed of up to 2.5 Gbps, direct agent communication may take less than a tenth of a second, and communications for decision making no more than a few seconds. Also, in its current form, the proposed agent-based TST does not take pedestrians into consideration. Given that close to 90% of Richardson's population commutes by either driving alone or carpooling, it is reasonable to assume that current pedestrian signal operations may not need to be modified. Nevertheless, we plan to incorporate pedestrian signal timing in future versions of our agent-based model.

References

1. Abdoos, M., Mozayani, N., & Bazzan, A. L. C. (2013). Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence*, *26*(5–6), 1575–1587.
2. Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, *129*(3), 278–285.
3. Al-Zinati, M., & Zalila-Wenkstern, R. (2015). Matisse 2.0: A large-scale multi-agent simulation system for agent-based its. In *IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology (WI-IAT), 2015* (vol. 2, pp. 328–335).
4. Balke, K, N., & Herrick, C. (2004). Potential measures of assessing signal timing performance using existing technologies. Technical Report FHWA/TX-04/0-4422-1, Texas A&M Transportation Institute, College Station, Texas 77843-3135.
5. Bazzan, A. L. C. (2005). A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, *10*(1), 131–164.

6. Bazzan, A. L. C., de Oliveira, D., & da Silva, B. C. (2010). Learning in groups of traffic signals. *Engineering Applications of Artificial Intelligence*, *23*(4), 560–568.
7. Bi, Y., Srinivasan, D., Xiaobo, L., Sun, Z., & Zeng, W. (2014). Type-2 fuzzy multi-intersection traffic signal control with differential evolution optimization. *Expert Systems with Applications*, *41*(16), 7338–7349.
8. Chao, K.-H., Lee, R.-H., & Wang, M.-H. (2008). An intelligent traffic light control based on extension neural network. In *Knowledge-based intelligent information and engineering systems* (pp. 17–24). Berlin: Springer.
9. Cheng, S.-F., Epelman, M. A., & Smith, R. L. (2006). Cosign: A parallel algorithm for coordinated traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, *7*(4), 551–564.
10. Chin, Y. K., Lee, L. K., Bolong, N., Yang, S. S., & Teo, K. T. K. (2011). Exploring Q-learning optimization in traffic signal timing plan management. In *2011 third international conference on computational intelligence, communication systems and networks* (pp. 269–274).
11. Collotta, M., Bello, L. L., & Pau, G. (2015). A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers. *Expert Systems with Applications*, *42*(13), 5403–5415.
12. Diakaki, C., Papageorgiou, M., & Aboudolas, K. (2002). A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, *10*(2), 183–195.
13. Dusparic, I., & Cahill, V. (2012). Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *7*(1), 11.
14. Dusparic, I., Monteil, J., & Cahill, V. (2016). Towards autonomic urban traffic control with collaborative multi-policy reinforcement learning. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)* (pp. 2065–2070).
15. El-Tantawy, S., & Abdulhai, B. (2010). An agent-based learning towards decentralized and coordinated traffic signal control. In *2010 13th international IEEE conference on intelligent transportation systems (ITSC)* (pp. 665–670).
16. El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, *14*(3), 1140–1150.
17. France, J., & Ghorbani, A. A. (2003). A multiagent system for optimizing urban traffic. In *Proceedings of the IEEE/WIC international conference on intelligent agent technology* (pp. 411–414).
18. Gartner, N. H., Pooran, F. J., & Andrews, C. M. (2001). Implementation of the OPAC adaptive control strategy in a traffic signal network. In *Proceedings. 2001 IEEE intelligent transportation systems, 2001* (pp. 195–200).
19. Goldstein, R., & Smith, S. F. (2018). Expressive real-time intersection scheduling. In *Thirty-second AAAI conference on artificial intelligence*.
20. Henry, J.-J., Farges, J. L., & Tuffal, J. (1983). The prodyn real time traffic algorithm. *IFAC Proceedings Volumes*, *16*(4), 305–310.
21. Hu, H.-C., & Smith, S. F. (2018). Bi-directional information exchange in decentralized schedule-driven traffic control. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems*. Bi-directional information exchange in decentralized schedule-driven traffic control (pp. 1962–1964).
22. Koonce, P., et al. (2008). Traffic signal timing manual. Technical report, United States. Federal Highway Administration.
23. Lu, S., Liu, X., & Dai, S. (2008). Incremental multistep Q-learning for adaptive traffic signal control based on delay minimization strategy. In *7th world congress on intelligent control and automation, WCICA* (pp. 2854–2858). IEEE.
24. Mannion, P., Duggan, J., & Howley, E. (2015). *Parallel reinforcement learning for traffic signal control* (Vol. 52, pp. 956–961). Amsterdam: Elsevier.
25. Mirchandani, P., & Head, L. (2001). A real-time traffic signal control system: Architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, *9*(6), 415–432.
26. Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., & Wang, Y. (2003). Review of road traffic control strategies. *Proceedings of the IEEE*, *91*(12), 2043–2067.
27. Pham, T. T., Brys, T., Taylor, M. E., Brys, T., Drugan, M. M., Bosman, P. A., Cock, M.-D., Lazar, C., Demarchi, L., & Steenhoff, D., et al. (2013). Learning coordinated traffic light control. In *Proceedings of the adaptive and learning agents workshop (at AAMAS-13)* (vol. 10, pp. 1196–1201).
28. Prabuchandran, K. J., AN, H. K, & Bhatnagar, S. (2014). Multi-agent reinforcement learning for traffic signal control. In *2014 IEEE 17th international conference on intelligent transportation systems (ITSC)* (pp. 2529–2534).
29. PTV Group. Ptv vissim. http://vision-traffic.ptvgroup.com/en-us/products/ptvvissim/. Accessed Oct 2018.

30. Richter, S. (2006). Learning traffic control-towards practical traffic control using policy gradients. *Albert-Ludwigs-Universitat Freiburg, Tech. Rep*.
31. Roads and Maritime Services. Scats: The benchmark in urban traffic control. http://www.scats.com.au/. Accessed Oct 2018.
32. Roozemond, D. A. (2001). Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research*, *131*(2), 293–301.
33. Smith, S. F., Barlow, G. J., Xie, X.-F., & Rubinstein, Z. B. (2013). Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system. In *Twenty-third international conference on automated planning and scheduling*.
34. Srinivasan, D., Choy, M. C., & Cheu, R. L. (2006). Neural networks for real-time traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, *7*(3), 261–272.
35. Torabi, B. (2017). A self-organizing traffic management system and its real-world implementation, Ph.D. proposal. Technical report, University of Texas at Dallas.
36. Torabi, Behnam, Al-Zinati, Mohammad, & Wenkstern, Rym Z. (2018). MATISSE 3.0: A Large-Scale Multi-agent Simulation System for Intelligent Transportation Systems. In *Proceedings of the 16th international conference on practical applications of agents and multi-agent systems, PAAMS 18*,Toledo, Spain (pp. 357–360).
37. Torabi, B., Wenkstern, R. Z., & Al-Zinati, M. (2018). An agent-based micro-simulator for ITS. In *Proceedings of the 21st IEEE international conference on intelligent transportation systems, IEEE ITSC 2018*, Maui, Hawaii, USA.
38. Torabi, B., Wenkstern, R. Z., & Saylor, R. (2018). A collaborative agent-based traffic signal system for highly dynamic traffic conditions. In *Proceedings of the 21st IEEE international conference on intelligent transportation systems, IEEE ITSC 2018*, Maui, Hawaii, USA.
39. Torabi, B., Wenkstern, R. Z., & Saylor, R. (2018). A multi-hop agent-based traffic signal timing system for the city of richardson. In *Proceedings of the the sixteenth international conference on autonomous agent and multiagent systems, AAMAS*, Stockholm, Sweden (pp. 2094–2096).
40. UK's Transport Research Laboratory. Split cycle and offset optimisation technique. https://trlsoftware.co.uk/products/traffic_control/scoot. Accessed Oct 2018.
41. UK's Transport Research Laboratory. Traffic network and isolated intersection study tool. https://trlsoftware.co.uk/products/junction_signal_design/transyt. Accessed Oct 2018.
42. Wen, K., Qu, S., & Zhang, Y. (2007). A stochastic adaptive control model for isolated intersections. In *IEEE international conference on robotics and biomimetics, 2007. ROBIO* (pp. 2256–2260).
43. Xie, X.-F., Smith, S. F., Chen, T.-W., & Barlow, G. J. (2014). Real-time traffic control for sustainable urban living. In *17th international IEEE conference on intelligent transportation systems (ITSC)* (pp. 1863–1868).