POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Distributed V2V-Based Virtual Traffic Light System

Original
A Distributed V2V-Based Virtual Traffic Light System / Rapelli, M., Casetti, C., Sgarbi, M.. - (2020), pp. 122-128. (2020 International Conference on COMmunication Systems and NETworkS, COMSNETS 2020 Bengaluru, India 2020) [10.1109/COMSNETS48256.2020.9027339].

Availability:
This version is available at: 11583/2822232 since: 2020-05-11T20:04:29Z

Publisher:
Institute of Electrical and Electronics Engineers Inc.

Published
DOI:10.1109/COMSNETS48256.2020.9027339

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

13 October 2026

A Distributed V2V-based Virtual Traffic Light System

Marco Rapelli
FULL Interdepartmental Center
Politecnico di Torino
Turin, Italy

Claudio Casetti
FULL Interdepartmental Center
Politecnico di Torino
Turin, Italy

Marcello Sgarbi
Politecnico di Torino
Turin, Italy

***Abstract*—Congestion in urban areas carries inherent costs in terms of fuel consumption, pollution, delays. Although ITS systems have devised solutions such as GLOSA, they require sophisticated infrastructure and thus installation and maintenance investments. In this paper, we propose V^{3}TL, a V2V-based distributed solution to manage unregulated intersections that aims primarily at minimizing the time needed to clear the intersection, while reducing the number of stop-and-go maneuvers that are known to increase emissions. The proposed solution operates a cyclic scheduling of vehicles as they approach the intersection, letting them coordinate among each other through V2V communication. We simulated the solution in an unregulated four-way intersection, showing that it achieves its objectives.**

***Index Terms*—ITS, Traffic Lights, V2V communication**

I. Introduction

It is universally acknowledged that cars are one of the leading causes of air pollution in urban areas, with serious negative effects on environment and health. Among the pollutants, particulate matter (PM10), carbon monoxide and ground-level ozone are related to a statistical increase in respiratory issues as well strokes and other cardiovascular diseases. Other emissions, such as CO_2, have been identified as the culprit of the greenhouse effect resulting in global warming. Although air pollution is not the only fallback of traffic congestion (commuters daily invest hours of their own time to negotiate the long queues entering or exiting a city during peak hours), it is by far the direst one. While the manufacturing of hybrid and electric cars is going to alleviate the problem in the long run, congested cities are in need of additional solutions.

It is to be remarked that fossil-fuel motor vehicles do not pollute just because they are on the road. The emission of pollutants is dependent on how the vehicles are used, and the driver has the chance to influence it through its behavior at the wheel such as driving at constant speed, accelerating smoothly and limiting the number of stop-and-go manoeuvres [1], [2]. Admittedly, the drivers are not always at fault for these behaviors: a vehicle waiting at an intersection is stopping and starting as those ahead of it clear the intersection and a new red light forces the remaining vehicles to stop again. The problem is compounded by the fact that, in many cases, the traffic lights timing does not match the real traffic conditions, forcing longer waiting times onto more congested travel directions. Intelligent traffic light systems realised by coordinating multiple networked traffic lights are in operation in many cities around the world. These solutions, however, lack any interaction with vehicles on and around the intersection.

In recent years, ITS (Intelligent Transportation Systems) devoted much effort to making our roads more secure, efficient and eco-friendly. Attempting to reduce the emission at intersections, the Travolution project [4] introduced the Green Light Optimal Speed Advisory (GLOSA) systems. The goal of the GLOSA service is to predict the green phases of traffic lights as well as to inform the drivers whether they can pass the traffic light within the present green phase. GLOSA provides information about traffic light signal phases by broadcasting messages to vehicles approaching an intersection. [3] showed through simulation that GLOSA can reduce fuel consumption by up to 22% for a single vehicle and around 8% in case of more vehicles on the road, and many works in the literature have come to similar conclusions. The biggest drawback of GLOSA is that it requires both traffic lights and vehicles equipped with a GLOSA box and a suitable vehicular communication technology (such as IEEE WAVE or ETSI ITS-G5). Also, it cannot be applied to unregulated intersections, lacking a traffic light because its deployment is deemed unnecessary or not cost-effective.

In this paper we introduce V^{3}TL, V2V-based Virtual Traffic Light, a distributed, dynamic solution that works without infrastructure, in an unregulated intersection. Through repeated scheduling cycles on a finite horizon, V^{3}TL aims at minimizing the crossing time at intersections while reducing the number of stop-and-go manoeuvres by vehicles. Our work improves the existing literature discussed in Section II by:

- allowing leaders to compute a convergent solution to the problem of intersection clearing with minimum delay and in a distributed fashion;
- reducing the number of stop-and-go manoeuvers;
- managing an unregulated intersection without the need of infrastructure;
- introducing a simplified, yet effective, intersection map and scheduling protocol that is amenable to being exchanged over the air using few bytes.

The rest of the paper is organized as follows: after a review of related work in Section II, Section III provides an overview of the system and the procedures need to implement V^{3}TL. A description of the simulation scenario and the performance

evaluation results can be found in Section IV, while Section V concludes the paper and discusses future work.

II. Related work

Our work contributes to an already rich literature on Virtual Traffic Lights, dating back to the seminal work by Dresner and Stone [5], which introduced reservation-based system applied to a simplified version of a real-world intersection traffic where cars were not allowed to turn and travelled at the same speed. Gradinescu et al. [6] introduced car-to-car communication to implement an adaptive traffic light system that however still requires controller nodes (and thus infrastructure was needed along with it) deployed at intersections. Among the first to devise a truly infrastructure-less virtual traffic light system, Ferreira et al. [7], [8] worked on a solution that leverages beacons exchanged by vehicles at an intersection and implements a leader-based message exchange aimed at informing all approaching vehicles of the current phase of the VTL at the intersection (without optimizing the phases). Through simulations, they also showed that VTL has the potential to reduce emissions by up to 18%. Building on the concept introduced in [7], Hagenauer, Sommer et al. [9], [10] improved the algorithms and protocols needed for the leader election. In a similar vein, Bazzi et al. [11] implemented a leader-based VTL on a real system using open software and low cost IEEE 802.11p devices.

III. Procedural description of the V^{3}TL System

The scenario under consideration consists of vehicles approaching an unregulated intersection. In order to simplify the description, a four-way intersection is assumed, although the methodology can be applied to other types of intersection. No roadside unit is assumed, therefore vehicles have to manage their interaction via information broadcasting in a V2V communication environment. The purpose is to create a distributed scheduling process that mimicks a virtual traffic light system with the goal of shortening the waiting time at the intersection. Therefore, each vehicle has to learn the presence of all other cars in the scenario. Since the purpose of our system is not to regulate *any* intersection at *any* time, the VTL is activated as soon as N_c vehicles are queueing up (not moving) in any one direction, which is taken as the sign that congestion is building. When this happens, a selected group of vehicles runs the scheduling process with a full awareness of all other vehicles and can thus act with a centralized knowledge. This group of vehicles will then distribute the output of the scheduling process to all other vehicles. The distributed scheduling process is arranged in *cycles*, during which the information on up to N_c vehicles per street of the intersection is collected and a suitable solution to clear the intersection of all the vehicles in the cycle is found.

A. Vehicle information collection

The information collection is divided into four steps as in Figure 1.

The first step of each cycle consists in allowing all approaching cars to identify other vehicles in the neighborhood of the intersection, including their anonymized data stored in an internal memory called *Vehicle Storage* and discovering their own position with respect to the crossroad, which we assume to be localized by its GPS coordinates.

The Vehicle Storage is filled by the information extracted from messages broadcasted by each vehicle. The messages exchanged to achieve this reciprocal knowledge are of BSM (Basic Safety Message) type, according to the SAE J2735 standard [13], and are broadcast by each vehicle with a frequency of 10Hz (i.e., the standard frequency mandated by IEEE). The information disseminated through BSMs is comprehensive of the basic motion characteristic of the vehicle, extended with some fields necessary for our purpose. In summary, the considered fields are the following:

- Vehicle anonymous random identification number
- Transmission timestamp
- Vehicle position (absolute and relative to the next intersection)
- Vehicle motion information, such as: speed, acceleration, heading, yaw angle, signaling lights (turns and breaks)
- Vehicle size
- Leader String (explained below)
- Intersection String (explained below)
- Solution Dataset (explained below)
- Intersection Flag (to notify if the intersection has been crossed)
- Scheduled Flag (to notify if a vehicle has already been considered in the scheduling process)
- Leader Election Flag

Upon a BSM reception, vehicles extract the corresponding fields from the message and store the information regarding the anonymous random identification number and the position of the message sender. In this way, a Vehicle Storage of all known vehicles is created by every vehicle and information is kept updated every time a BSM with fresh data is received. It is important that a Vehicle Storage also has the vehicle's own position in it (self location). For this reason, upon a BSM reception, the vehicle's own location is updated, while the transmitter information is saved in the Vehicle Storage. The first step of the cycle is preliminary to the prosecution of the algorithm: when a vehicle in any direction detects that at least N_c vehicles including itself are queueing up in its direction, it sets the Leader Election flag in its BSM, triggering the following steps of the vehicles information collection.

The message exchange phase allows the second step of the cycle, i.e, the identification of the so-called *Direction Leader (DL)*, to begin. Due to the possible presence of buildings in the environment, vehicles likely receive messages from vehicles in other directions only in proximity of the intersection, while they can easily store data from those in their same direction. While knowledge of vehicles in each direction is likely to be complete, the same cannot be said about vehicles in other directions. A "leader" vehicle for every direction thus takes it

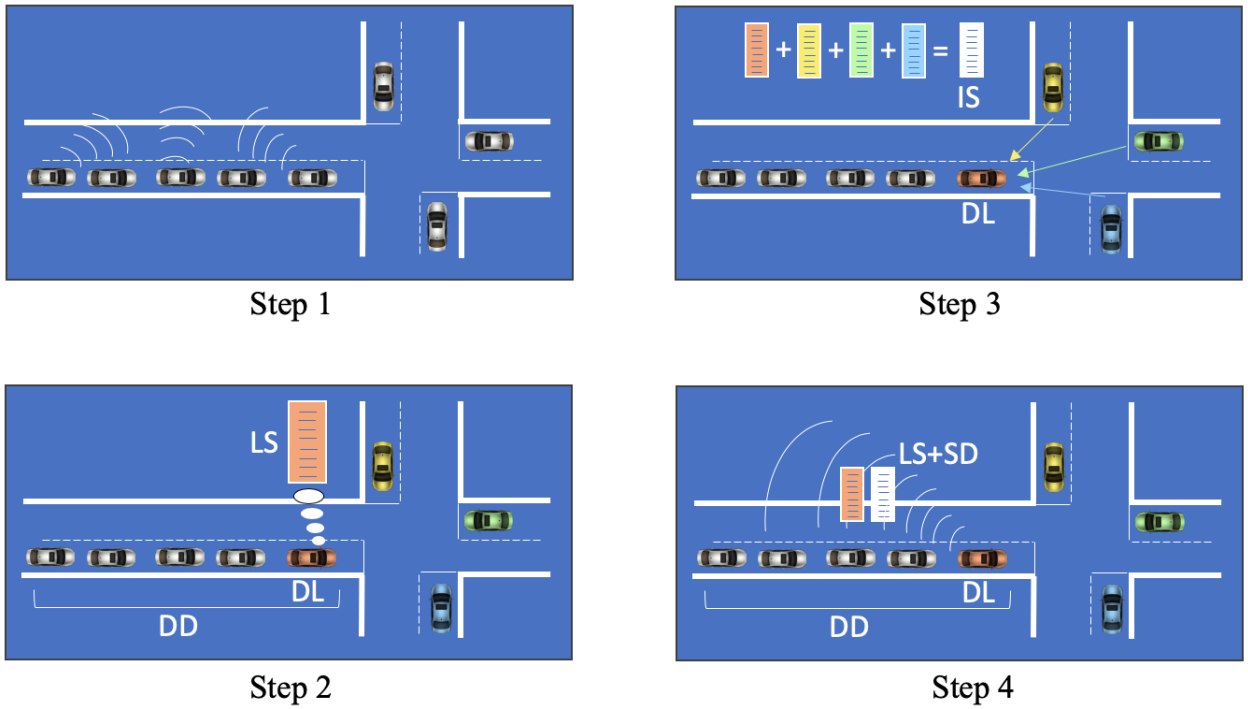

Fig. 1. Steps of information collection in an cycle. Step 1: vehicles exchange BSMs; Step 2: Leader election and Leader String compilation; Step 3: Direction Leaders exchange Leader Strings, then merge them into the Intersection String; Step 4: Each DL broadcasts the Leader String and the Solution Dataset to the rest of the vehicles

upon itself to collect the data and exchange it with other leaders in other streets. A vehicle that has not yet been scheduled in a previous cycle is elected as DL if it is the vehicle closest to the intersection in its respective direction when the Leader Election procedure is triggered. During the message exchange, each DL collects the information on vehicles approaching the junction from its own direction. Those vehicles become part of the *Direction Dataset (DD)* of the corresponding DL and their motion information are saved in a *Leader String (LS)*. If those vehicles have declared a signaling light (or, in its absence, the intention of going straight on), such information is extracted and included in the Leader String. Thus, for every vehicle of the list, the information regarding, e.g., identification number, position in the queue, signaling light and direction (eastbound, southbound, westbound or northbound) is saved. In each cycle, the information of up to N_c cars per direction is collected, which means that the depth of each Leader String is at least N_c (thus including the Leader's own position). The Leader String is ordered by growing distance from the intersection. The DL occasionally has to account for the situation where, on its street, fewer than N_c cars have reached the intersection at the time the vehicle information collection took place. In this case, for every empty position, a "ghost" vehicle is generated in that slot and included by the DL in its LS as a stopping vehicle, devoid of declared motion intentions. Creating such a structure largely simplifies the VTL scheduling, indeed it is possible to work considering only N_c "tiers" of vehicles. The vehicles of the first tier will be the Leaders, the four cars directly behind them will form the second tier, and so on. Algorithm 1 shows a pseudocode summarizing the workflow of the DL. If a vehicle is not elected DL, it just saves the identifier of its own DL and awaits scheduling instructions from its own DL.

In the third step, the composition of the information from the N_c tiers forms the *Intersection Dataset (ID)*, which is the complete set on which the scheduling for the cycle is performed. In practice, the ID is created when the DLs approach the intersection: upon entering radio-visibility of each other, they exhange their Leader Strings in order to merge their DDs and create the Intersection Dataset. Leaders Strings are thus merged in a single *Intersection String (IS)*.

In the fourth and final step, the IS is used by each DL to compute the scheduling solution, called *Solution Dataset (SD)*, as explained in the following subsection. Since the DLs use as input the same IS, they will nominally output the same SD. Upon completion of this task, each DL broadcasts its own LS (to univocally assign the tier position to all cars in its DD) and the SD (to distributed the scheduling instructions to all tier positions in its direction). Therefore, a vehicle can abide to the scheduling process only when it has knowledge of the complete intersection status through the received LS and SD. This also guarantees that the same scheduling solution is executed by each vehicle. Finally, it allows every vehicle to update its Vehicle Storage, marking those vehicles that have been scheduled in the current cycle (and will thus be no longer around in the next cycle). Any vehicle that, at the time of the

```
Data: On BSM received
Update my info in the Vehicle Storage;
Extract info of the transmitter from BSM and add it to
  the Vehicle Storage;
Compute DistanceToIntersection for every vehicle in
  vehicle storage;
Select DLs based on the minimum
  DistanceToIntersection per direction;
Save the identification number of my own DL and
  check if I am the DL;
if I am Direction Leader then
    foreach Vehicle in my Direction Set do
        if That vehicle is not scheduled then
            if That vehicle has a declared turn
              intention then
                Extract its info from Vehicle Storage;
                Add info to Leader String;
                Order Leader String by ascending
                  DistanceToIntersection;
            end
        end
    end
    if A new Leader String has been received then
        Merge Leader Strings;
    end
    if All Leader Strings have been received then
        Create the Intersection String;
        Run SCHEDULER from Intersection String
          and derive the Solution Dataset;
        Broadcast Leader String and Solution Dataset;
    end
end
```

Algorithm 1: Direction Leader workflow

leader election, occupied a position more than N_c vehicles away from the intersection, will not find its identification number in the broadcasted LS and will not be scheduled. It will then trigger and participate in the Leader Election procedure and information collection of the next cycles.

B. Scheduling solution computation

The vehicles obviously need a compact representation of the intersection and of its occupants to run the scheduling algorithm. Also, the Solution Dataset requires another, similarly compact representation to be broadcasted effectively. We chose to represent the intersection as a bitmap, thus dividing it into small cells and each of them is modeled as a bit, as can be seen in Figure 2. In such a representation, the zeros correspond to a position that could be occupied by a vehicle, while the ones are the locations forbidden or already occupied by other vehicles. The intersection area, in particular, is represented as a 3x3 bitmap. This is the minimum number of bits that does not result in ambiguities in crossing vehicle positions or in forbidden turns.

Fig. 2. Bitmap representation of the intersection: queue position in red, intersection area in black.

Based on this scheme, a function to inspect all the legal movements was implemented. It works on an Intersection Dataset composed by N_c tiers and, for every tier, it selects all the possible configurations resulting from the movement of vehicles at the top of the queue in each direction, taking into account all the possibilities: turn right, turn left, go straight or do not move. It then discards any illegal configuration where two or more vehicles overlap. Thanks to this approach, we can limit the analysis of the possibile scheduling solutions to legal configurations only, thereby greatly reducing the computational complexity. Namely, for the case we studied in this paper ($N_c = 6$ in a four-way intersection), we only had to analyze 64,800 legal configurations out of $(4 \cdot 6)^4 = 331,776$. The goal of this scheduling is to minimize the number of actions needed to empty the intersection. An "action" is defined as a "legal" movement, i.e., when one or more vehicles simultaneously cross the intersection without conflicting with any other vehicle. A secondary goal is to minimize the number of stop-and-gos performed by each vehicle in order to reduce emissions. Figure 3 shows an example of three legal actions.

It is conceivable that, for a small value of N_c, one could use a brute-force approach computing all possible solutions and evaluating the best one in terms of smaller number of actions needed to clear the intersection by all cars in the cycle. The outputs of the brute-force approach for different N_c and for different intersections configurations (e.g., three-, four-, five-way intersections and so on) might also be pre-loaded in each vehicle, or broadcasted by an RSU or through the cellular network for all intersections of a certain area. However, we propose a low-complexity heuristics to address a general case

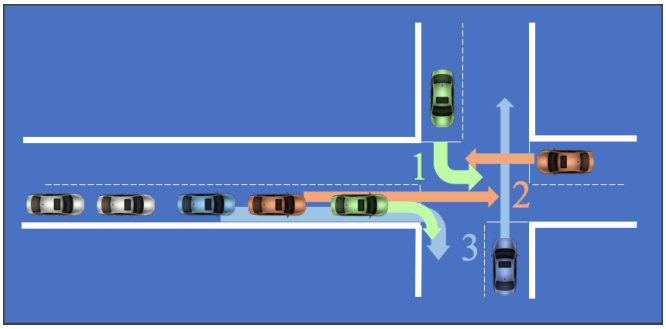

Fig. 3. Example of three legal actions: action 1 (green) allows the head-of-the line vehicles from the southbound and the eastbound street to move simultaneously, followed by action 2 (orange) which dictates that another eastbound vehicle crosses, together with the westbound one; eventually, action 3 (blue) allows the northbound vehicle to move together with another eastbound one turning right.

(recall that the scheduling must be computed in real time by the Direction Leaders).

Given a starting configuration provided by the Intersection String with $M \leq 4N_c$ vehicles at the intersection, the scheduling algorithm first determines all the I legal actions that involve the vehicles of the first tier, $\{A_{i,1}\}$. For each $A_{i,1}$, with $i = 1, .., I$ an integer metric $c(A_{i,1}) \in [1, 4]$ is computed as the number of cars that clear the intersection as a result of the action. From this starting point, one solution tree is built for each root $A_{i,1}$, looking at what actions can follow $A_{i,1}$. In order to keep the complexity low, every level of the tree can add at most B actions to each node of the previous level.

Next, for each $A_{i,1}$, all additional singular legal actions are evaluated, and B of them are selected as $A_{i,j,2}$ with $j = 1, .., B$. The selection criteria aims at maximizing the respective $c(A_{i,j,2})$ computed as the number of vehicles clearing the intersection as a result of $A_{i,1}$ and the action being considered. As a tie-breaker, the number of stop-and-gos generated by the solution, $s(A_{i,j,2})$, is taken. We count as stop-and-go a situation where in the previous action a vehicle moved and, in the current action, the vehicle right behind it was not allowed to move. In case of a lingering tie, a random choice among the best solutions is made. In such a way, the second level of the solution tree is built and we record the path of the tree with the highest value of c and lowest s as the current scheduling choice.

The third level is built repeating the previous step for the action set $\{A_{i,j,2}\}$ and so on. After completing a level, we prune any branch whose metric c compared to that of the current scheduling choice solution reveals a gap higher than the unscheduled cars in the intersection for that branch.

The output of the function is the *Solution Dataset*, which contains as many rows as there are legal actions in the solution. It is structured as a four-column matrix where each row has the form:

$$\{\{C_e, A_e\}\{C_s, A_s\}\{C_w, A_w\}\{C_n, A_n\}\}$$

where C refers to the order of the tier the vehicle belongs to, i.e., its position in the queue *at the time of the scheduling*, while A refers to one of four possible coded instruction: (1) brake and stop; (2) turn right; (3) turn left; (4) go straight. The pedices refer to the direction to reach the intersection (eastbound, southbound, westbound, northbound).

For example, with reference to the sequence of actions illustrated in Figure 3, the first three rows of the scheduled solution would be represented as follows:

$$\{\{1,2\}\{1,3\}\{1,1\}\{1,1\}\}$$
$$\{\{2,4\}\{2,1\}\{1,4\}\{1,1\}\}$$
$$\{\{3,2\}\{2,1\}\{2,1\}\{1,4\}\}$$

Such a solution mandates (action 1, row 1) that the vehicles from the eastbound and the southbound streets move simultaneously, respectively turning right and left without hindering each other; instead, the northbound and westbound vehicle should stop as the action is carried out. Next, (action 2, row 2) another eastbound vehicle (in position 2 at the time of the scheduling) goes straight, like the westbound one, while the northbound still has a red light; it is to be remarked that a "ghost" vehicle shows up in the solution in the second column from now on, which is nothing more than a placeholder at this point (indeed, no other vehicles are coming southbound). Finally, (action 3, row 3), the northbound vehicle goes straight and a third eastbound vehicle is allowed to turn right; another ghost vehicle is shown in the third column (westbound), as a consequence of having cleared that direction with the previous action.

IV. Results

The results shown in this section were obtained from simulations, where every vehicle is modeled explicitly with its own movements and its own OBU in charge of managing the exchange of messages. In order to develop such a complex scenario, a simulation framework based on Veins [12] was developed. Veins bridges the contribution of two other open-source simulators, SUMO (Simulator of Urban Mobility), used to model the mobility environment and OMNeT++, which is the network simulator. SUMO and Veins exchange information via TCP socket, using the TraCI (Traffic Control Interface) communication protocol. In this way vehicle movements defined by SUMO are reflected in OMNeT++ and actions triggered by messages can be actuated in SUMO.

In order to evaluate the performance of the implemented model, we considered a four-way intersection, where a version of our model with $N_c = 6$ tiers and with a binary tree ($B = 2$) scheduling heuristics was tested against two other scenarios without VTL. In each case, the total time needed to empty our map was collected and the outputs of ten simulations were averaged, obtaining a final time measure. We have included buildings on all sides of the intersection, hindering the communication among different streets.

First of all, in the spirit of addressing our original goal of allowing unregulated intersections to benefit from a VTL, a scenario with such an intersection was tested. In SUMO, when a junction type is defined as unregulated, the "right-before-left" priority scheme is adopted by approaching vehicles. In the

second scenario, a traffic light with fixed phases is involved. Since the default generation of a traffic light in SUMO creates non-realistic control phases, a traffic light designed with F. V. Webster method [14] was used. The Webster method is a mathematical technique currently used by civil engineers in order to design optimal timing phases, given a realistic incoming traffic. Its purpose is to compute the optimum cycle length C and the corresponding green times G_i, given:

- the maximum number of vehicles that can pass the intersection in one hour, called saturation flow S;
- the lost time L, which is the total time lost due to human reaction times for every cycle;
- the critical flow rate y_i, computed as the ratio between the number of vehicles per direction v_i and the saturation flow S.

Once those parameters are defined, the following formulas are needed to compute C and G_i:

$$C = \frac{1.5 \cdot L + 5}{1 - \sum_i y_i} \tag{1}$$

$$G_i = \frac{(C - L) \cdot y_i}{\sum_i y_i} \tag{2}$$

In order to estimate the saturation flow of the intersection in the model, a continuous flow of vehicles has been generated in SUMO. We set all vehicles as wanting to go straight at the intersection. Finally, an average of 1174 veh/h was measured and used as the saturation flow S. A time loss of 2s, due to reaction times, was estimated for every green phase, resulting in a total time lost L of 4s. Regarding the number of vehicles per direction, needed to compute the critical flow rate y_i, the flows generated in our scenario do not represent realistic traffic data. For this reason, we used vehicle data from J. He et al. [15], where real traffic measurements are extracted from a four-way intersection similar to the one used in this paper. The flows per direction and the corresponding y_i used are the following:

$$v_e = 470 \rightarrow y_e = \frac{470}{1174} = 0.4 \tag{3}$$

$$v_s = 203 \rightarrow y_s = \frac{203}{1174} = 0.17 \tag{4}$$

$$v_w = 137 \rightarrow y_w = \frac{137}{1174} = 0.12 \tag{5}$$

$$v_n = 364 \rightarrow y_n = \frac{364}{1174} = 0.31 \tag{6}$$

Since the northbound and southbound directions will share the same green phase, as well as eastbound and westbound ones, only the maximum critical flow rate per direction has been considered, resulting in:

$$y_{n \wedge s} = 0.31 \tag{7}$$

$$y_{e \wedge w} = 0.4 \tag{8}$$

From the parameters obtained, the resulting optimal cycle length is:

$$C = \frac{1.5 \cdot 4 + 5}{1 - (0.31 + 0.4)} = 37.9s \tag{9}$$

Fig. 4. Traffic light phases computed with Webster method.

TABLE I
INTERSECTION CLEARING TIME COMPARISON

Simulation Number	Unregolated Intersection [s]	Webster Traffic Light [s]	V^{3}TL [s]
1	197.2	167	165.2
2	173.6	177.6	167.6
3	187.2	179	174.6
4	187.7	193	177.1
5	192.2	163.6	171.7
6	183.9	174.7	179.1
7	200.3	174	168.2
8	178.6	195.7	180.5
9	179.4	194.3	179.2
10	183	193.1	178
Average ⇒	186.3	181.2	174.1

and the corresponding green phases are:

$$G_{n \wedge s} = \frac{(37.9 - 4) \cdot 0.31}{0.31 + 0.4} = 14.8s \tag{10}$$

$$G_{e \wedge w} = \frac{(37.9 - 4) \cdot 0.4}{0.31 + 0.4} = 19.1s \tag{11}$$

To complete the cycle, an amber phase of 1s after every green phase and a clearance interval with both phases set as red for 2s have been added, as shown in Figure 4, and such a traffic light and its phases have been generated in SUMO in order to create the Webster scenario.

In order to compare the results of different scenarios, ten simulations were performed, each featuring 66 vehicles. In each simulation, the intentions of each vehicle as it approached the intersection were randomly selected, thus determining different solutions in each test. The time to solve the total configuration of vehicles, for every scenario, is reported in Table I.

As shown in the table, not only does V^{3}TL perform better on average, but it also outperforms both right-before-left scenario and traffic light with the Webster method in almost every single test. It is indeed to be remarked that in tests 5 and 6 the Webster method fares better than V^{3}TL. This occurs because it allowed situations in SUMO that V^{3}TL does not contemplate. Namely, vehicles can stop in the middle of the intersection and let other vehicles pass if continuing with the crossing leads to a collision. These types of vehicle configuration lead to speeding up the process of clearing the intersection and result in shorter time to solve the queueing. However, they can create hazardous situations and were thus not considered by the scheduling.

V. Conclusions and future work

The paper proposes V^{3}TL, a V2V-based distributed solution realizing a Virtual Traffic Light system for unregulated intersections. By going through scheduling cycles, we define a heuristics that aims at lowering the time needed to clear the intersection, while limiting the number of pollution-increasing starts and stops. Simulation shows that our solution outperforms both a right-before-left scenario and a fixed traffic light whose cycle is optimized with the Webster method in almost every single test. As future work, we plan to adapt our solution to different intersection topologies and roundabouts, and to extend the methodology to multiple neighboring intersections, with the potential to carry over the signaling between each of them and introducing a "green wave" effect.

Acknowledgment

The authors would like to thank FEV Italy s.r.l. for contributing to preliminary discussions on this paper.

References

[1] J. Cloke, G. Harris, S. Latham, A. Quimby and E. Smith, "Reducing the environmental impact of driving: a review of training and in-vehicle technologies," Report 384, Transport Research Lab, UK, 1999.

[2] M. Andre and U. Hammarstrom, "Driving speeds in Europe for pollutant emissions estimation," Transportation Research, Part D, vol. 5, pp. 321-335, 2000.

[3] T.Tielert, M.Killat, H.Hartenstein, R.Luz, S.Hausberger and T.Benz, "The impact of traffic-light-to-vehicle communication on fuel consumption and emissions," in Internet of Things, Tokyo, Japan, Dec 2010.

[4] R. Braun, F. Busch, C. Kemper, R. Hildebrandt, F. Weichenmeier, C. Menig, I. Paulus and R. Presslein-Lehle, "Travolution – Netzweite Optimierung der Lichtsignalsteuerung und LSA-Fahrzeug-Kommunikation," Strassenverkehrstechnik, vol. 53, pp. 365–374, June 2009.

[5] K. Dresner and P. Stone, "Multiagent Traffic Management: A Reservation-Based Intersection Control. Mechanism," in 3rd International Joint Conference on Autonomous Agents and Multiagent Systems. (AAMAS 2004). New York, NY: IEEE, Jul. 2004, pp. 530–537.

[6] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea, and L. Iftode, "Adaptive Traffic Lights. Using Car-to-Car Communication," in 65th IEEE Vehicular Technology Conference (VTC2007-Spring), Apr. 2007, pp. 21–25.

[7] M. Ferreira, R. Fernandes, H. Conceicao, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control,". in 7th ACM International Workshop on Vehicular Internetworking (VANET 2010). Chicago, IL: ACM, Sep. 2010, pp. 85–90.

[8] M. Ferreira and P. d'Orey, "On the Impact of Virtual Traffic Lights on Carbon Emissions Mitigation,". IEEE Transactions on Intelligent Transportation Systems, vol. 99, no. 10-2011, pp. 1–12, Oct. 2011.

[9] F. Hagenauer, P. Baldemaier, F. Dressler, and C. Sommer, "Advanced Leader Election for Virtual. Traffic Lights," ZTE Communications, Special Issue on VANET. 12. 11-16, 2014.

[10] C. Sommer, F. Hagenauer, and F. Dressler, "A Networking Perspective on Self-Organizing Intersection. Management," in IEEE World Forum on Internet of Things (WF-IoT 2014). Seoul, Mar. 2014.

[11] A. Bazzi, A. Zanella, B. Masini, "A Distributed Virtual Traffic Light Algorithm Exploiting Short Range. V2V Communications," Ad Hoc Networks, Vol. 49, pp. 42-57, October 2016.

[12] veins.car2x.org [online].

[13] Society of Automotive Engineers, SAE-J2735, Dedicated Short Range Communications (DSRC) Message Set Dictionary, 2015.

[14] F. V. Webster, "Traffic signal settings," Gt. Britain Road Res. Lab., vol. Road Resea, 1958.

[15] J. He and Z. Hou, "Ant colony algorithm for traffic signal timing optimization," Adv. Eng. Softw., vol. 43, no. 1, pp. 14–18, 2012.