# A Distributed Algorithm for Adaptive Traffic Lights Control in Wireless Sensor Networks

Sébastien Faye, Claude Chaudet, Isabelle Demeure

***Abstract*— In this paper, we address the problem of controlling traffic lights at an intersection with a wireless sensor network. We propose a wireless sensor network architecture that does not depend on a centralized coordinator and we separate logically this distributed network into 4 levels of hierarchy. On this architecture, we define and evaluate through simulation a phases selection algorithm that decides dynamically of the movements composing the next phase and of the duration of this phase. Simulation results show that this algorithm, if properly tuned, has the capacity to reduce average waiting time at an intersection, while avoiding starvation for multiple load levels.**

## I. Introduction

Traffic Lights Controllers (TLC) are devices that define a road intersection behavior by controlling when each traffic light becomes red or green and for how long. These devices traditionally use a static plan: switching sequence and timings are pre-determined and are independent of the traffic conditions. Adapting the order and duration of the green lights in function of the actual traffic could prevent for example leaving a green light when no vehicles wants or is able to cross the intersection. Such reactive strategies shall improve significantly the road network performance, reducing the traffic load as well as the users journey time.

In this article, we explore how to use a wireless sensor network (WSN) to dynamically control traffic lights. WSN are a kind of ad-hoc network in which elements have limited capacities in terms of energy, memory, computation power and communication. These sensor nodes, which communicate using wireless transmissions, represent a cheap alternative to the classical induction loops and can therefore be deployed at a much higher density, potentially controlling every intersection of a city.

Classically, the literature addresses intersections that are composed of four directions, as represented on figure 1. Each direction is further decomposed into one left lane for vehicles turning left and one or more right lanes for vehicles going straight or turning right. A TLC controls, at each moment, which **movements** are allowed. Each movement is usually identified and represented by the cardinal directions of its origin and destination. For example, on figure 1, *WE* denotes the movement from West to East[1].

At a given intersection, multiple movements can occur simultaneously, provided that they do not interfere. Such a combination of movements is called a **phase**. A sequence of phases in which every movement is selected at least once is called a **cycle**.

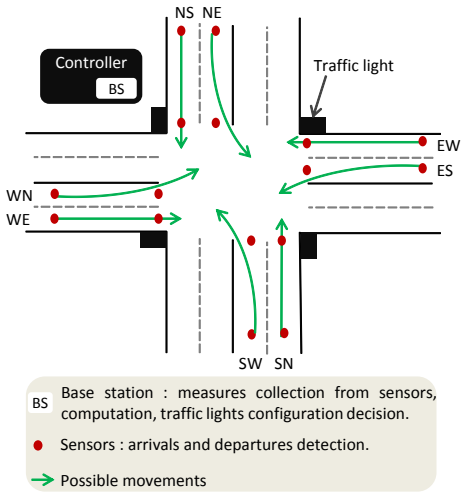



Fig. 1: A typical 4-lanes intersection with 2 sensors per lane.

The work presented here aims at letting a WSN dynamically compose phases based on its perception and on the data individual sensors exchange. Section II reviews relevant related works in Intelligent Transport Systems (ITS) and WSN domains. We then specify a hierarchical WSN architecture in section III and propose a traffic lights control algorithm at a single intersection in section IV. We evaluate this algorithm by simulation in section V. The results we obtain demonstrate that an adaptive control algorithm leads to a smaller average waiting time than a fixed, predetermined scheme. We conclude the paper in section VI and discuss the extension of this work to multiple intersections.

## II. Related Works

### A. Wireless Sensor Networks for ITS

Adaptive ITS that use sensors ([1], [2], [3], [4], [5]) are generally used to feed a queueing model, which requires to evaluate either the number of vehicles on each lane of an intersection, or to capture the vehicle arrival process intensity. If radars and induction loops are typically used for such measurement, their cost reserves them to main roads. Magnetometers represent much cheaper alternatives to count of the number of vehicles in an area or on a lane ([6], [7], [8]). They work similarly as induction loop, detecting metallic vehicles by measuring the change on the earth's magnetic field. According to [7], such a device is able to

All authors are with the Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI UMR 5141, Paris, France (e-mails: {first.last}@telecom-paristech.fr).

[1]The NEMA (*National Electrical Manufacturers Association*) numbering provides a more general, numerical nomenclature that allows to represent more complex intersections.

detect 99% of the vehicles that pass over it. Two coupled magnetometers can identify a vehicle type and measure its speed and length if they are separated by a known distance. Cameras represent an even cheaper solutions, as they do not require roadworks for installation and can achieve a fair accuracy with image processing techniques, even though they have a limited angle of vision and are sensitive to obstruction.

Such sensors can be improved with an attached processing unit to perform image processing or to filter out false detections. They can also be given communication capabilities for remote management and maintenance through a wireless network interface that does not require cabling. With appropriate power source (battery, solar panel or wiring on the traffic light, e.g.), these devices are ready to form a WSN able to monitor every intersection in a neighborhood or a city and every lane of each intersection.

The use of such a network reporting measurements and receiving directives from a control center has already been evoked in the literature. Mimbela *et al.* [8] and Knaian [9] both evoke a very low manufacturing cost – less than $ 30 per unit – and small size – similar to a coin – and hence confirm that sensors can be efficient in many environments. Corredor *et al.* [10] even show that such networks perform better than induction loops, because of their responsiveness, ease of installation and number of measurement points.

### B. Network architecture

As illustrated on figure 1, a typical sensor network for ITS is generally deployed around a traffic controller, or base station, that provides at least access to a global network and hence connectivity to a control center in which operators are able to modify the lights behavior and timing. Such an ITS generally comprises multiple sensors deployed on the road. The question of the number and position of the sensors is important, as it influences the measurement quality and defines the core of the network architecture. Monitoring every circulation lane with a sufficient accuracy requires to deploy either one magnetometer per lane, or to have a $360^o$ coverage of the intersection with cameras. Zou *et al.* ([5]) use only one sensor per direction and assume this sensor is able to detect vehicles up to a distance of 5 meters, and to distinguish two lanes for each direction.

Using a second sensor on every lane allows to reach a better accuracy ([3]). In addition, it allows to detect abnormal behaviors that a single sensor may miss, for example frauds or inactivity of a vehicle. Coupling both sensors also allows to evaluate other metrics such as vehicles speed or lengths.

Concerning the sensors placement, it is possible to pre-calculate their positions and to install them statically, or to allow dynamic placement in function of the traffic condition. Using a too small distance makes the system inefficient, as the measured queues size are quickly bounded. Yousef *et al.* [2] propose to separate the two sensors by a distance that corresponds to 8 vehicles, while Tubaishat *et al.* [3] prefer a shorted distance, between 5 and 8 vehicles. Zhou *et al.* [4] estimate that the best distance between two sensors is equal to the product of the average vehicle speed by the maximum time a light is allowed to stay green and hence that this distance should be dynamic.

### C. Traffic lights control algorithms

The contributions evoked above suppose that the sensor network is only used to report measurements to a central server that takes decisions globally. However, a WSN disposes of a certain computation power and could implement local algorithms, solving easy problems without the help of a central decision point. Such a local approach not only enhances responsiveness, as communication latency is lower, but also fault tolerance, as the failure of the base station, for instance, does not cut sensors from all intelligence anymore. A few authors have examined how such an autonomous intersection could work. The core notations used in this section and in the rest of this paper are listed in table I.

| | |
|---|---|
| $T_G$ | Duration a given light remains green. |
| $T_s$ | Vehicles start-up delay when a light becomes green. |
| $T_{max}$ | maximum time a light is allowed to stay green (limited by operators, usually to enhance users perception of the network responsiveness). |
| $N_i$, $N^y$ | Queue length corresponding to lane $i$ or to movement $y$. |
| $\lambda$ | Average vehicles arrival frequency. |
| $\mu$ | Average frequency at which vehicles leave the intersection when the light is green (departure rate) |

TABLE I: Notations used in this article

Al Nasser *et al.* [1] compute an average queue length for each lane and define green time as $T_G = min(T_s+\Delta, T_{max})$, where $\Delta$ is a variable time that depends on this queue length. Yousef *et al.* [2] propose a more developed solution. In figure 1 eight possible movements exist : two per incoming direction. The main idea of this paper is to describe each movement $y$ as an M/M/1 queue. The different movements queues lengths ($N^y$) and the average waiting time ($AWT = N^y/\lambda$) are determined using Little's law. If we denote by $T_G$ the time a light stays green and by $T_R$ the time it stays red, the queue length for a lane $i$ varies according to $N_i^C = N_i^{C-1} + \lambda T_G - \mu T_G + \lambda T_R$, where $C$ represents the current cycle number and $\mu$ the average departure frequency. $\lambda T_G$ and $\lambda T_R$ vehicles arriving during the green and red light respectively and $\mu_G T_G$ vehicles leave during the green light. Using this equation and a matrix that identifies conflicting movements, the algorithm proposed by Yousef *et al.* selects movements combinations in order to minimize the average queue length and waiting time. The algorithm determines all allowed movements combinations, sums the number of vehicles in the corresponding queues, and select, as the next phase, the movements set that has the largest total number of vehicles. The green light time is then calculated proportionally to the queues sizes.

Tubaishat *et al.* propose a general model in [11] and provide a complete traffic lights control solution in [3]. They pre-define three set of phases, composed respectively of 4, 6 and 8 phases. Cycles are then defined on one of these sets by ordering phases in a greedy manner based on the queues

sizes. Evaluation is performed with Green Light District simulator (GLD, [12]), which does not support variable green light times, though. These contributions suppose a conflict-free scheduling and are therefore too rigid in several situations. In addition, considering only the queues length may lead to famine situations, as in [2].

Zhou et al. ([4]) provide a traffic lights plan based on movements combinations that can be performed simultaneously without any conflict. For example, on figure 1, EW and WE movements can happen simultaneously, as well as WN and WE, or WN and ES, which defines 8 phases possibilities. Their algorithm then selects the sequence of phases in a cycle, according to the following criteria, by decreasing importance :

1) Lanes with imperatives (e.g. : emergency vehicles).
2) The number and size of *blanks*. A blank is defined as an event that is raised each time no vehicle is detected by a sensor during a time $T_{Blank} > L_{veh}/V_{veh}$. The blank is valued with a length $L_{Blank} = T_{Blank} * V_{veh}$. If all combinations have at least one blank, the one having the shortest first recorded blank is selected. If this combination is selected, more cars can pass. If one or more cases have no blank, then we pass the next test.
3) Each lane hunger level, to prevent starvation.
4) The combination that has the largest total waiting time.
5) The largest queue.

Finally, Zou *et al.* [5] define green lights times using fuzzy logic : each green time is determined based on a set of load intervals. For example, if less than five vehicles by minute are detected, the green time will be 10 seconds.

## III. WSN ARCHITECTURE

In our work, we chose to suppose that the sensors used to monitor the vehicles traffic are magnetometers, as they are accurate and (relatively) cheap. This only influences the proposed deployment and does not represent a strong constraint, though. Cameras could represent a better alternative, especially when it comes to installation-related civil works, as they can be installed on the light directly, without any roadworks. However, they are more easily obstructed and pose privacy issues.

Based on the results and good practices from the literature, a sensor network that monitors and controls an intersection should be composed of at least two magnetometer sensors per lane. The distance between sensors should be sufficient to have a correct sampling. Ideally the distance between the two sensors should be dynamic, which can be achieved either by placing a moving detector on security rails, or by multiplying the installed sensors and selecting the best couple of active sensors in function of the traffic condition. These sensors shall collect, aggregate and exchange data in order allow selecting a phases that will be communicated to the TLC that is in charge of changing the green lights accordingly.

The TLC only plays the role of the actuator in this scenario and computation of the light plan can be performed by any node. Similarly, the base station role is limited to the one of a simple communication interface, providing access to the control center that can disseminate global policies and directives. Such a link also provides access to other sensors that can exchange data about other intersections, preventing for instance, cascade effects. The TLC and base station can be located on the same physical machine, or separate depending on the local setup.

Direct communication from the sensor devices to the base station prevents spatial reuse, which could lead to a wireless channel capacity problem when the network becomes dense. Sensors should therefore form a low-range mutlihop network and auto-organize. Auto-organization has been widely studied in the fields of wireless mutihop networks and autonomous solutions exist for addresses allocation, election of central points and routing. Ad-hoc and sensors routing protocols, in particular, are mature enough so that we can safely rely on an off-the-shelf solution to create and maintain routes, reacting to nodes faults or wireless channel problems.

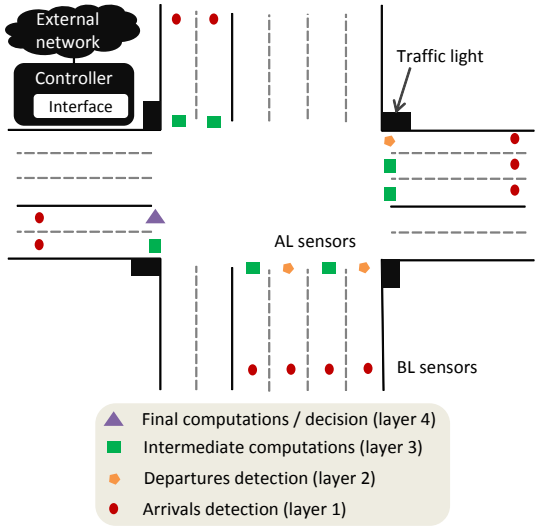



Fig. 2: Our hierarchical model

In this scenario, we chose to organize sensors in a hierarchical architecture, as represented on figure 2. Sensors are organized in two main layers : (1) Before Light (BL) sensors continuously collect vehicle arrivals, and are placed at a distance chosen by the designer; (2) After Light (AL) sensors collect departures, only when the corresponding light is green. AL sensors have less load to handle than BL sensors and consequently, they are in charge of data aggregation and decision-making process. We may further divide the set of AL sensors in two, defining an additional layer : in case a movement involves several lanes, we must elect a sensor that aggregates collected data for each movement. Finally, we have to elect a master sensor, that collects each movement data and applies a decision algorithm. This sensor only needs to inform AL sensors and to transmit the corresponding order to the interface, that transmits to TLC for decision application. Figure 3 represents this hierarchy and materializes communication paths.

This architecture does not give particular roles to individ-

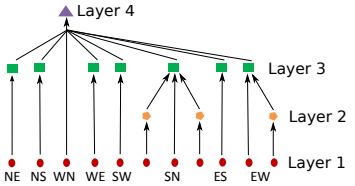


Fig. 3: Sensors hierarchy and communication paths

ual sensors. The sensors that belong to the highest layers (layer 3 and 4) are elected among the set of AL sensors, and they can be re-elected when the control center decides so, or when the sensors themselves notice a neighbor's failure, radio channel overload or any other problem.

A battery-operated sensor (e.g. a mobile sensor) may also detect when its battery level decreases under a certain threshold and send a re-election request to its neighbors. When a layer-1 or layer-2 sensor is faulty, the layer-3 sensors can either fall back on a redundant sensor if available, or use statistical or pre-defined data in place of the acquired information.

This hierarchical architecture also eases data aggregation : each layer naturally aggregates data from the lower layer. Arrivals can be detected and accumulated by BL sensors over a full phase and results can be transmitted to AL sensors only once per phase, which saves energy and bandwidth. Finally, AL sensors may sleep when red light triggers.

## IV. Traffic lights control algorithm

### A. Philosophy

We use the architecture described above as the supporting infrastructure for a traffic lights control algorithm. This algorithm is designed to be flexible and easily adaptable to any intersection configuration. Even though it takes decisions on its own, at the level of a single infrastructure, it can be customized or influenced by engineers and operators that can set variables from the control center. More specifically, operators can specify the desired behavior of each intersection by uploading the set of allowed simultaneous movements through the *conflict matrix* described below, or tune user-level parameters such as the maximum waiting time allowed, $T_{max}$. If the classical algorithms usually work at the cycle granularity, we chose to have a more reactive approach. Instead of defining cycles, we re-evaluate the situation at every phase and select the next phase based on the observed system parameters. The notion of cycle does not exist anymore in our model.

### B. Conflicts management

Our algorithm uses a conflict matrix, that describes all possibles cases of conflicting movements and drives phases creation. In practice, some intersections allow certain conflicts to reduce the number of possible phases. In this case, green light is given to low priority movements simultaneously with higher priority movements. We consider two possibilities, to study the algorithm behavior in two different cases : either the conflict matrix forbids all simultaneous movements as soon as an interaction exists (such as matrix #1 on figure 4), or certain conflicts that do not pose safety problems are allowed (matrix #2 on figure 4). Matrix #1 does not only allow or forbid certain simultaneous movements, it also keeps track of which movements are in conflict, which allows an algorithm to treat differently the case in which a movement is selected alone and the case in which it is selected with a conflicting movement. It is also necessary to represent single-lane systems, e.g. when vehicles turning left can block vehicles going straight. The matrices here only record allowed and disallowed conflicts, but a larger scale can be used to represent different conflict severities.

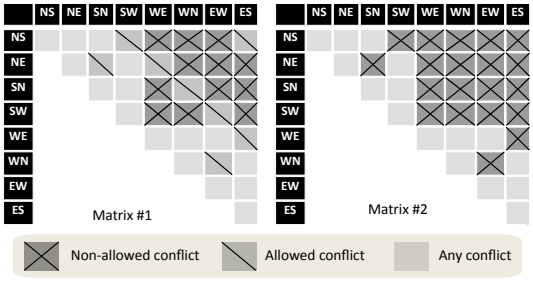


Fig. 4: Conflict matrices

*1) General algorithm on a single intersection:* Once the architecture is in place and configuration data such as the conflict matrix is obtained from the control center, the different sensors start to communicate during phase $P$ in order to select dynamically the which movements will compose phase $P+1$. The algorithm then apply is composed of 7 steps described hereafter.

1) **Count**: For each lane $i$, each BL sensor sends the number of arrivals during the phase P ($N_i^A$) to its corresponding AL sensor and resets its vehicle counter to 0. Each AL sensor monitors the number of vehicles departures during the phase ($N_i^D$) and keeps track of the number of vehicles that were present on the lane at the beginning of the previous phase ($N_i^P$). From these values, it computes the number of vehicles at the beginning of the phase P+1: $N_i^{P+1} = N_i^P + N_i^A - N_i^D$. If others lanes are used for the same movement, it transmits $N_i^{P+1}$ to the movement layer 3 leader sensor.
2) **Movement aggregation**: Each movement leader, $y$, maintain the time elapsed since the last selection of the movement, $T_F^y$, to detect and prevent starvation. It sums the $N_i^{P+1}$ values to get $N^y$, the total queue length for movement $y$. Finally, it transmits these two values to the network leader (layer 4) sensor.
3) **Evaluation**: Layer 4 leader computes the score function ($S(y)$) for each movement $y$ according to the following algorithm that takes into account famine and queue length:
   a) If no vehicle is present for movement $y$ (i.e. $N^y = 0$), $S(y) = 0$.
   b) Otherwise, $S(y)$ is computed by summing $T_F^y$ and $N^y$. Each of these values is normalized and weighted by user-defined weights ($W_N$ and

$W_{T_F}$), in order to let operators favor one or the other objective. For an intersection that comprises $M$ movements, the score of a given movement is defined by:

$$S(y) = W_N \cdot \left( N^y \cdot \frac{100}{\sum_{a=1}^{M} N^a} \right) + W_{T_F} \cdot \left( T_F^y \cdot \frac{100}{\sum_{a=1}^{M} T_F^a} \right)$$

This expression mixes starvation and queues length-related criteria, which leads to good results in several situations. It cab however easily be modified to include data coming from other intersections, policies transmitted from a control center, or to limit explicitly starvation time.

4) **Candidate phases listing**: depending on the conflict matrix, layer 4 leader computes all combinations of conflict-free movements. If some conflicting movements are allowed, they are added up, possibly with a reduced influence, or by bounding for example the maximum number of cars allowed/expected to pass in this case.
5) **Phase selection**: among the set of combinations (each candidate phase), select the combination with maximum total score. At this stage, additional criteria can be considered (e.g. : emergency vehicle detection, combination avoiding in case of accident detection).
6) **Define green light time**: once the phase is selected, the minimum time required to let all vehicles pass is equal to $T_P = T_s + N_{max} * T_H$, where $N_{max}$ is the number of vehicles for the lane having the greatest number of vehicles among all selected phase lanes. For simulations, we set $T_s = 4$ and $T_H = 2$, complying with [13], but these timings can be learnt or adjusted during the network lifetime. Letting all vehicles of one lane pass can lead to an excessive waiting time for other lanes. We need to bound this time by a static value, $T_{max}$. In the case of $T_P < T_{max}$, additional vehicles can arrive on AL sensors, and make the green time increase by $T_H$, until eventually reaching $T_{max}$.
7) **Application**: finally, layer 4 leader sends the result to the interface for application. Moreover, it broadcasts a reset message to layer 3 sensors concerned by the phase, so that $F^y = 0$ and $T_F^y = 0$ for the next phase. All sensors continue logging arrivals and departures. When the phase is finished, the light stays green and the algorithm is re-executed. Thus, in some cases, lanes can keep green light. In other cases, yellow light starts for an estimated duration of 4 seconds ([13]).

### C. Transmission costs evaluation

A naive communication protocol in which sensors only report to a central decision point their measurement results would generate *total arrivals + total departures* notifications per phase. With our communication protocol, there are between *3*total lanes - 1* and *3*total lanes - 1 + additional vehicles* transmissions per phase, without the leaders election and self-organization protocols. Self organization protocols usually rely on regular broadcast of control frames and election can be performed in $O(log(\alpha))$ messages, where $\alpha$ is the number of nodes to elect.

### D. Algorithm extensions

Here, we focus on traffic lights control; however, extensions can easily be envisioned : collision risk detection, control center influence, add pedestrian management, lanes individualization (bus or taxi network management), etc. Moreover, we can extend this algorithm to multiple intersections. In this case, each intersection executes its own algorithm with its own parameters, but can anticipate vehicles arrivals from neighboring intersections. Consequently, we can imagine a new $S(y)$ influence parameter, growing gradually as more vehicles approach. We do not explore this approach here. Finally, insist that $S(y)$ is central to the phase selection selection phase : we can use extra weights to favor other objectives, for example influence the user to take particular directions.

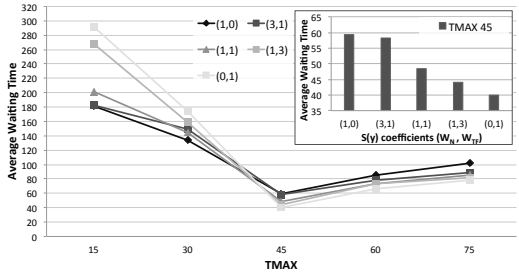
## V. Simulations

In this section we evaluate our algorithm with SUMO (Simulation of Urban MObility, [14]). SUMO is an open-source, discrete time, continuous space and microscopic simulator entirely coded in C++ to model traffic flow. Specifically, it allow to place sensors and retrieve their values by connecting to a simulation, and allows to create TLC algorithms. We evaluated the previously described algorithm in five load repartition scenarios : 1, 2 (opposite or orthogonal), 3 or 4 directions are subdued to a low arrival rate ($\lambda$ vehicles per second) while the others suffer from a higher load ($3.\lambda$ vehicles per second). We evaluated each of these scenarios for three different base traffic values ($\lambda = \lambda_0$, $\lambda = 3.\lambda_0$, $\lambda = 6.\lambda_0$) and each of the simulations ran during 2000 program steps, which represents 2000 s. The results presented below are the average on 5 runs of a scenario (repartition, load) with different random seeds.

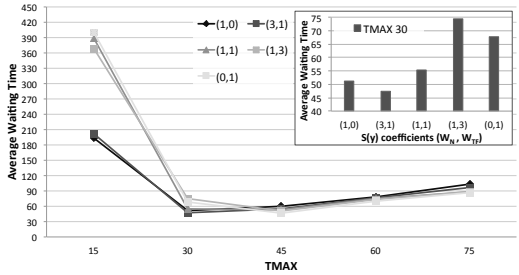
### A. Parameters choice

The choice of the scoring function weights ($W_N$, $W_{T_F}$) and of the green time limit ($T_{max}$) are expected to have a strong influence on the performance result. We chose to evaluate these parameters by testing 5 weights configurations ($W_N$, $W_{T_F}$): (1,0), (3,1), (1,1), (1,3), (0,1) and five values for $T_{max}$: 15, 30, 45, 60, 75 seconds. Because of space constraints, we only present the results for the uniform high load scenario. However, all results exhibit the same behavior, and all our results are available online at [15].

Figures 5(a) and 5(b) represent the average waiting time at an intersection for the different parameters combinations, when conflicts are forbidden and allowed respectively. The inline figure represents, a zoom on the average waiting times for the different weights combination for $T_{max}$ value that gives the best performance. We can first notice that




(a) ($W_N$,$W_{T_F}$) and $T_{max}$ influence, conflicts allowed




(b) ($W_N$,$W_{T_F}$) and $T_{max}$ influence, conflicts forbidden

Fig. 5: Average waiting time for scenario 5

allowing conflicting movements allows to reach a better average waiting time (40 s vs. 47 s in the represented setup). On these figures, we can notice that the best $T_{max}$ value is different when conflicts are allowed ($45\,s$) or forbidden ($30\,s$). When conflicts are forbidden, letting all movements happen requires more phases, which leaves less time to a single phase. Results on the different weighting configurations show that the weights configuration also depends on whether conflicts are allowed or not. In the former case, famine reduction should be favored, while in the latter case it is queues lengths that should receive the higher weight.

### *B. Performances*

Finally, for both matrices, using the best values obtained, we compare results with a SUMO predetermined light plan. This plan was selected among 4 others, explained and detailed at [15]. Figure 6 shows that the adaptive strategy remains the most efficient, especially when the traffic increases. Moreover, a strategy that allows movement in conflict is more effective as long as we properly dimension $T_{max}$.

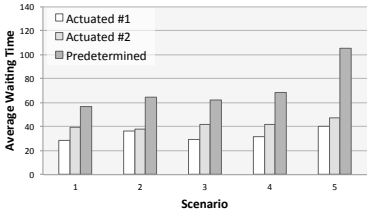



Fig. 6: Performance results

## VI. Conclusion and Future Works

In this paper, we proposed a distributed algorithm to control traffic lights in urban areas. We introduced a new model that allowed us to avoid the use of a central point (BS) for manage locally the intersection, to distribute overhead costs and that is easy to establish. The proposed solution is flexible in conflicts management and performs more frequent decisions than presented works (one by phase instead of one by cycle). Our results provide some clues on adjusting the algorithm parameters, and show a high efficiency compared to predetermined solutions.

In future work, our results encourage us to extend our distributed algorithm to the multiple intersections case, to explore how intersections can communicate between them, in a realistic and distributed manner, and what does this can bring. Also, introduce new elements, like public transport lanes or pedestrian crossing, could be interesting and would be closer to reality.

## References


[1] F. A. Al-Nasser and H. Rowaihy, "Simulation of dynamic traffic control system based on wireless sensor network," in *IEEE Symposium on Computers Informatics (ISCI)*, Kuala Lumpur, Malaysia, Mar. 2011.

[2] K. M. Yousef, J. N. Al-Karaki, and A. M. Shatnawi, "Intelligent traffic light flow control system using wireless sensors networks," *Journal of Information Science and Engineering*, vol. 26, no. 3, May 2010.

[3] M. Tubaishat, Q. Qi, Y. Shang, and H. Shi, "Wireless sensor-based traffic light control," in *5th IEEE Conference on Consumer Communications and Networking (CCNC 2008)*, Las Vegas, USA, Feb. 2008.

[4] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system," in *72nd IEEE Vehicular Technology Conference Fall (VTC 2010-Fall)*, Ottawa, Canada, Sep. 2010.

[5] F. Zou, B. Yang, and Y. Cao, "Traffic light control for a single intersection based on wireless sensor network," in *9th International Conference on Electronic Measurement & Instruments (ICEMI 2009)*, Beijing, China, Aug. 2009.

[6] S. Coleri, S. Y. Cheung, and P. Varaiya, "Sensor networks for monitoring traffic," in *In Allerton Conference on Communication, Control and Computing*, Monticello, USA, Sep. 2004.

[7] S. Cheung, S. Coleri, B. Dundar, S. Ganesh, C. Tan, and P. Varaiya, "Traffic measurement and vehicle classification with single magnetic sensor," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1917, no. -1, pp. 173–181, Dec. 2005.

[8] L. E. Y. Mimbela and L. A. Klein, *Summary of vehicle detection and surveillance technologies used in intelligent transportation systems*. Federal Highway Administration, Intelligent Transportation Systems Joint Program Office, 2007.

[9] A. N. Knaian, "A wireless sensor network for smart roadbeds and intelligent transportation systems," Master's thesis, Massachusetts Institute of Technology, Jun. 2000.

[10] I. Corredor, A. García, J. Martínez, and P. López, "Wireless sensor network-based system for measuring and monitoring road traffic," in *6th Collaborative Electronic Communications and eCommerce Technology and Research (CollECTeR 2008)*, Madrid, Spain, Jun. 2008.

[11] M. Tubaishat, Y. Shang, and H. Shi, "Adaptive traffic light control with wireless sensor networks," in *4th IEEE Conference on Consumer Communications and Networking*, Las Vegas, USA, Jan. 2007.

[12] M. Wiering, J. Vreeken, J. van Veenen, and A. Koopman, "Simulation and optimization of traffic in a city," in *Intelligent Vehicles Symposium, 2004 IEEE*, Jun. 2004, pp. 453 – 458.

[13] R. Gordon, W. Tighe, U. S. F. H. A. O. of Operations, D. E. Associates, and I. Siemens, *Traffic control systems handbook*. US Dept. of Transportation, Federal Highway Administration, Office of Operations, 2005, http://ops.fhwa.dot.gov/publications/fhwahop06006/.

[14] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *The Third International Conference on Advances in System Simulation (SIMUL 2011)*, Barcelona, Spain, Oct. 2011, pp. 63–68.

[15] http://perso.telecom-paristech.fr/~faye/ITSC2012/.