2012 15th International IEEE Conference on Intelligent Transportation Systems
Anchorage, Alaska, USA, September 16-19, 2012

A Distributed Algorithm for Adaptive Traffic Lights Control

Sébastien Faye, Claude Chaudet, Isabelle Demeure

Abstract— **In this paper, we address the problem of controlling traffic lights at an intersection with a spatially distributed sensor network. We propose a sensor network architecture that does not depend on a centralized coordinator and we separate logically it into 4 levels of hierarchy. On this architecture, we define and evaluate through simulations an adaptive traffic light control algorithm. Based on two main objectives, this algorithm decides dynamically of the green lights sequences by selecting the movements composing each phase and its duration. Simulation results show that this algorithm, if properly tuned, has the capacity to reduce average waiting time at an intersection, while avoiding starvation.**

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 paper, we define a **sensor** as a node that has a detection unit and generally limited capacities in terms of energy, memory, computation power and communication. Here, we explore how to use a spatially distributed sensor network to dynamically control traffic lights on an isolated intersection. This type of network is able to make decisions without external assistance and therefore be used to control one intersection or potentially every intersections 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**.

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 representing more complex intersections.

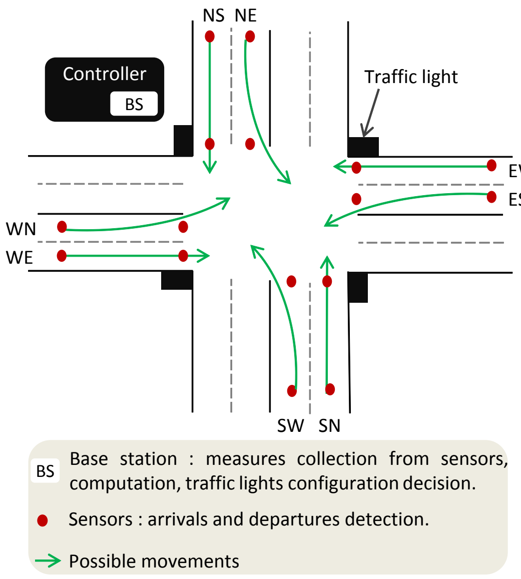

NS
NE
Controller
BS
Traffic light
EW
ES
WN
WE
SW
SN
BS Base station : measures collection from sensors, computation, traffic lights configuration decision.
● Sensors : arrivals and departures detection.
Possible movements

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

The work presented here aims at letting a sensor network 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 especially adaptive traffic lights control. We then specify a hierarchical sensor network architecture in section III and propose a traffic lights control algorithm at a single intersection in section IV. Based on an intersection of Amiens city (France), 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. Sensor Networks for ITS

In this paper, we chose to suppose that the sensing units used to detect vehicles are magnetometers, as they are accurate, small and (relatively) cheap ([1]). 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 roadwork. However, they are more easily obstructed and pose privacy issues. A magnetometer is traditionally attached to a wired or wireless sensor, which provides energy, memory, computation and communication units. Such sensors constitute a kind of ad-hoc network which can be placed precisely on each lane of an intersection and thus allows us to study, on a small-scale, how properly articulate an algorithm. Adaptive ITS using sensors ([2], [3], [4], [5]) generally feed a queuing model, which requires to evaluate 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. This type of detector, based on a magnetometer, is able to record a unique signature for 99% of vehicles ([6]), by measuring the changes on Earth's magnetic field when users pass over. Corredor *et al.* [7] even show that such networks perform better than induction loops, because of their responsiveness, ease of installation and number of measurement points. Knaian [1] evokes a very low manufacturing cost – less than $ 30 per unit – and small size – similar to a coin – and hence confirms that sensors can be efficient in many environments.

B. Network architecture

As illustrated on figure 1, a typical sensor network for ITS is generally deployed around a traffic controller, or base station, providing 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 ° coverage of the intersection with cameras.

Traditionally, it is considered that each incoming lane is equipped with two sensors: one located at the light to count departures and one other at a fixed distance before the light, to count arrivals. Generally, it is recommended to set this distance between 5 and 8 vehicles ([2], [3]) or depending on the maximum authorized green time ([4]). This model can use one sensor by lane (but results are less accurate) or one by direction (but the model is limited by the sensor detection range, according to [5]).

C. Traffic lights control algorithms

Some authors have examined how an autonomous intersection could work. The core notations used in this section and in the rest of this paper are listed in table I.

Yousef *et al.* ([2]) propose to describe each movement of an intersection as a M/M/1 queue and to select the most appropriate phases based on non-conflicting movements. 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 possible phases. 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

T_G	Duration a given light remains green.
T_s	Vehicles start-up delay when a light becomes green.
T_H	The average headway between discharging vehicles.
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.
λ	Average vehicles arrival frequency.
μ	Average frequency at which vehicles leave the intersection when the light is green (departure rate)

TABLE I: Notations used in this article

red, the queue length for a lane i varies according to $N_i^C = N_i^{C-1} + \lambda T_G - \mu T_G + \lambda T_R$. C represents the current cycle number, λT_G and λT_R vehicles arriving during the green and red light respectively and $\mu_G T_G$ vehicles leave during the green light. Using this equation, the algorithm proposed by Yousef *et al.* selects combinations of movements 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. In the same way, Tubaishat *et al.* ([3]) propose to define cycles by ordering three set of pre-defined phases, in a greedy manner based on queues sizes. These two 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. Their algorithm then selects the sequence of phases in a cycle, according to the following criteria: the presence of priority vehicles, length of periods where no vehicle is detected, starvation, total waiting time and queue length. This algorithm, however, is based on unrealistic assumptions, requiring the same vehicle type and speed.

These contributions are the closest to our work. However, they are sometimes unrealistic or based on pre-determined methods and suppose that the network is only used to report measurements to a controller that takes decisions. Nevertheless, a sensor dispose 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, but also fault tolerance, as the failure of the base station, for instance, does not cut sensors from all intelligence anymore.

Many others aspects of adaptive traffic lights control are developed in the literature. On the one hand, large-scale adaptive control systems can manage traffic policy of entire cities, like SCOOT [8] or SCATS [9]. While the former measures the traffic throughput in order to send a performance indication from a control center, the latter set lights plans based on a hierarchical architecture. These global systems require a global view to work: our contribution, as well as previous presented works, can make local decisions on one isolated intersection, potentially control every intersection of a city or come in addition to these systems. On the other hand, some traffic lights control algorithms based on artificial intelligence and others theoretical tools exist. Chen *et al.* ([10]) uses a genetic algorithm to optimize the time of green lights, approaching the solution by successive mutations. Zou *et al.* ([5]) define green lights times using fuzzy logic: the green time is selected depending on the counted vehicle number per minute. Some authors use a cooperative network, where vehicles are able to communicate with each other and with the traffic lights. Wenjie *et al.* ([11]) use a sensor network while Houli *et al.* ([12]) use a multi-agent system: a multi-objective decision is voted among the agents (vehicles). These contribution are generally purely theoretical and elusive. In the case of a cooperative network, it is hard to imagine, nowadays, all vehicles with an embedded sensor. Fuzzy logic is not sufficient to represent the real-time traffic uncertainty. Finally, the neural networks and specifically genetic algorithms needs many computations and their parameters are difficult to determine.

Our contribution, detailed below, provides an architecture and a distributed algorithm, easy to implement, realistic and does not rely on pre-determined methods.

III. Traffic lights control architecture

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. Here, we want to count the number of vehicles able to pass within T_{max} seconds: we propose to set $D = N * L_{veh}$. $N = \frac{T_{max}-T_s}{T_H}$ is the number of vehicles passing in theory in T_{max} seconds and L_{veh} is the average length of a vehicle. For simulations and for the rest of this paper, we set $T_s = 4$ and $T_H = 2$, complying with [13], but these timings can be learnt or adjusted during the network lifetime. 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 any node can perform computation of the light plan. 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.

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)

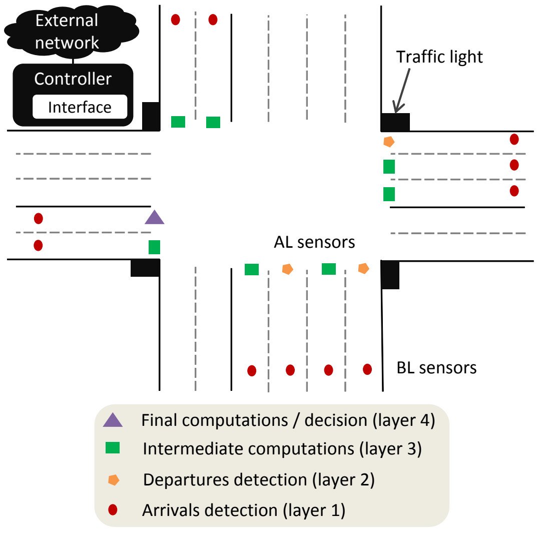

External network
Controller
Interface
Traffic light
AL sensors
BL sensors
Final computations / decision (layer 4)
Intermediate computations (layer 3)
Departures detection (layer 2)
Arrivals detection (layer 1)

Fig. 2: Our hierarchical model

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, which transmits to TLC for decision application. Figure 3 represents this hierarchy and materializes communication paths.

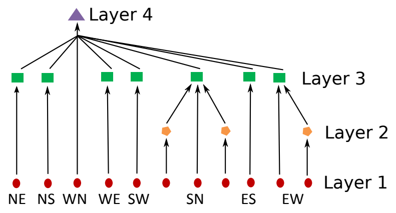


Fig. 3: Sensors hierarchy and communication paths

This architecture does not give particular roles to individual 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.

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 allowed green time, 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 possible 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. Conflicting movements can be joined to a phase as a non-priority or conditional movements, e.g. if there are less than a certain threshold of vehicles. Such a matrix 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. In some cases, such a matrix is used only during the network installation and can be called when changes happen on the intersection (e.g. roadwork).

C. 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

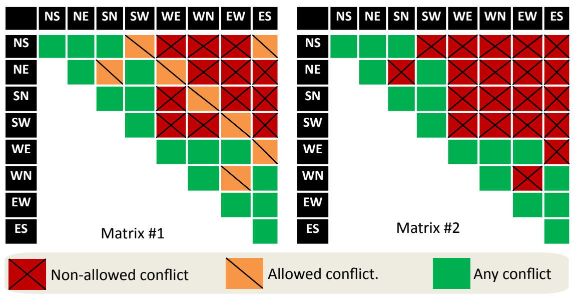


Fig. 4: Conflict matrices

compose phase $P+1$. The algorithm is composed of 7 steps described hereafter, according to the notations in table I.

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. The default values of these weights can be $W_N = W_{T_F} = 1$. However, engineers can redefine them according to the traffic knowledge. For an intersection that comprises M movements, the score of a given movement is defined by:

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

This expression mixes starvation and queues length-related criteria, which leads to good results in several situations. It can 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. A combination is a union of several movements that are authorized by the conflict matrix to have the green light in the same time. 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**: the selected phase P is the combination with the highest score sum. 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. Letting all vehicles of one lane pass can lead to an excessive waiting time for other lanes. We need to bind 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]).

D. 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 incoming lanes - 1* and *3*total incoming 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 α is the number of nodes to elect.

E. Algorithm extensions

Here, we focus on traffic lights control; however, extensions can easily be envisioned: collision risk detection, control center influence, addition of 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 process: 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 allows placing sensors and retrieving their values by connecting to a simulation, and allows creating TLC algorithms. We evaluated the previously described algorithm based on an intersection modeled from Amiens city (France), on which we have counting statistics and the lights plan used at a peak hour, between 8 am and 9 am. This allows us to evaluate our algorithm with a realistic traffic and compare it with an appropriate lights plan. This intersection has 4 directions and 12 possible movements: from each direction, a vehicle can go straight, turning left or turning right. Each simulation ran during 3600 program steps, which represents 3600 s. The results presented below are the average waiting time computed on the 3000 first vehicles, with an arrival rate $\lambda = 0.8$ new vehicles per second on the intersection.

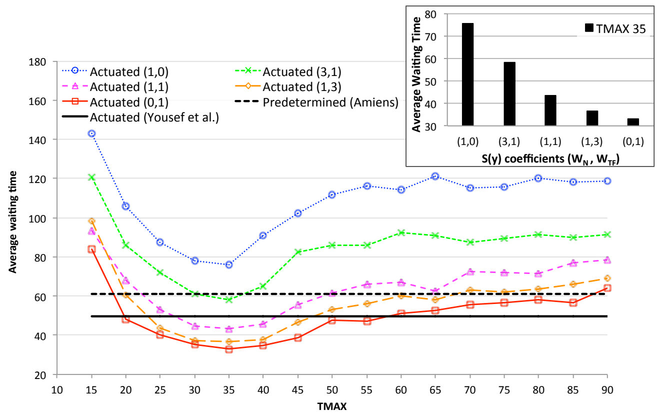
A. Parameters choice

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 five weights configurations (W_N, W_{T_F}): (1,0), (3,1), (1,1), (1,3), (0,1) and sixteen values for T_{max}: 15 to 90 seconds in steps of 5 seconds.

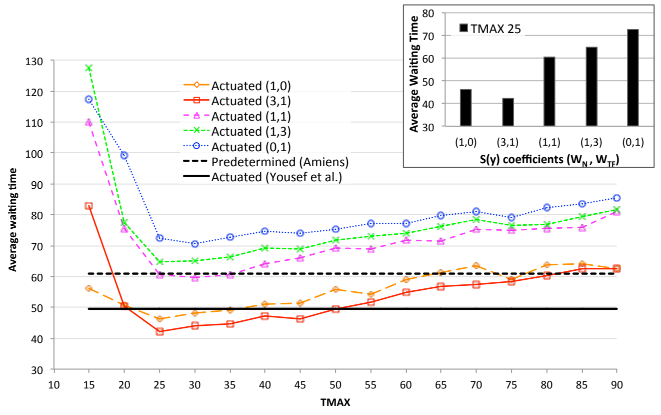
Figures 5(a) and 5(b) represent the average waiting time at an intersection for the different parameters combinations, when conflicts are forbidden and allowed respectively. The inline figure represents a zoom on the average waiting times for the different weights combination for T_{max} value that gives the best performance. We can first notice that allowing conflicting movements allows reaching a better average waiting time (28 % saving time in the represented setup). On these figures, we can notice that the best T_{max} value is different when conflicts are allowed ($35\,s$) or forbidden ($25\,s$). 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. Performance

Finally, for both conflict matrices, using the best values obtained, we compare results of the adaptive algorithm



(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: Performance results

presented above with the ones achieved by a predetermined lights plan designed by civil engineers and used in Amiens city and an adaptive algorithm based on Yousef *et al.* proposition ([2]), that is the work the closest to ours. As introduced above, the predetermined light plan is designed to respond to the traffic that we simulate.

Figures 5(a) and 5(b) show that the adaptive strategy (denoted by *actuated*) achieves the best result, especially when the traffic increases, provided that the correct value of T_{max} is selected. Moreover, our strategy is more efficient that an algorithm based on Yousef *et al.* proposition, due to starvation limitation and green time computation. Finally, we can see that allowing movement in conflict is more effective as long as we properly dimension T_{max}.

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.

Acknowledgments

The authors thank Thierry Delaporte, head of the traffic control center of Amiens, who provided the data needed by our simulations.

References

[1] A. N. Knaian, “A wireless sensor network for smart roadbeds and intelligent transportation systems,” Master’s thesis, Massachusetts Institute of Technology, Jun. 2000.

[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. 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.

[7] 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.

[8] D. Robertson and R. Bretherton, “Optimizing networks of traffic signals in real time-the scoot method,” *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 11 –15, Feb. 1991.

[9] A. Sims and K. Dobinson, “The sydney coordinated adaptive traffic (scat) system philosophy and benefits,” *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130 – 137, May 1980.

[10] X.-F. Chen and Z.-K. Shi, “Real-coded genetic algorithm for signal timing optimization of a single intersection,” in *2002 International Conference on Machine Learning and Cybernetics*, vol. 3, 2002, pp. 1245 – 1248.

[11] C. Wenjie, C. Lifeng, C. Zhanglong, and T. Shiliang, “A realtime dynamic traffic control system based on wireless sensor network,” in *International Conference Workshops on Parallel Processing (ICPP 2005)*, Jun. 2005, pp. 258 – 264.

[12] D. Houli, L. Zhiheng, and Z. Yi, “Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network,” *EURASIP J. Adv. Signal Process*, vol. 2010, pp. 7:1–7:7, Mar. 2010.

[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.