Multiple intersections adaptive traffic lights control using a wireless sensor networks

Contrôle adaptatif des feux de circulation sur de multiples intersections à l'aide d'un réseau de capteurs sans fil

Sébastien Faye
Claude Chaudet
Isabelle Demeure

2013D005

août 2013

Département Informatique et Réseaux
Groupe RMS : Réseaux, Mobilité et Services

Contrôle adaptatif des feux de circulation sur de multiples intersections à l'aide d'un réseau de capteurs sans fil

Sébastien Faye, Claude Chaudet, Isabelle Demeure
Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI UMR 5141, Paris, France
{*prenom.nom@telecom-paristech.fr*}

Août 2013

Abstract

Dans cet article, nous détaillons et étudions TAPIOCA (*distribuTed and AdaPtive IntersectiOns Control Algorithm*), un algorithme distribué qui défini les séquences et durées des feux verts dans un système de transport intelligent. TAPIOCA s'appuie sur les données recueillies par un réseau de capteurs sans fil déployé aux intersections afin de décider localement de son programme. TAPIOCA ne se reposant pas sur une entité centrale, il est réactif et facile à installer. Nous exposons tout d'abord une version simple intersection de TAPIOCA, qui a pour but d'améliorer la longueur des files d'attente et la probabilité de famine. Nous étendons ensuite TAPIOCA au cas des multiples intersections en définissant des mécanismes facilitant la circulation entre intersections proches afin de créer des vagues vertes. Les algorithmes sont évalués à l'aide du simulateur SUMO par le biais de trois scénarios: données réelles issues de la ville d'Amiens, le projet TAPASCologne et une grille d'intersections. Les résultats montrent que TAPIOCA génère un temps moyen d'attente inférieur à d'autres stratégies, adaptatives ou non.

Multiple Intersections Adaptive Traffic Lights Control using a Wireless Sensor Networks

Abstract

In this paper, we detail and study TAPIOCA (*distribuTed and AdaPtive IntersectiOns Control Algorithm*), a distributed algorithm to define the green light sequences and durations in an urban intelligent transportation system. TAPIOCA relies on data gathered by a hierarchical wireless sensors and actuators network deployed at intersections to decide locally of an intersection schedule. As TAPIOCA does not rely on a central entity, it is responsive and easy to install. We first expose a single-intersection version of TAPIOCA, which aims at improving the queues lengths and the starvation probability and then extend TAPIOCA to the multi-intersection case by defining mechanisms to ease offloading between close intersections and to create green waves. Both algorithms are evaluated with the SUMO simulator in three scenarios: real data from Amiens, the TAPASCologne project, and a grid. The results show that TAPIOCA achieves a low average waiting time compared to other dynamic strategies and to a fixed schedule pre-determined by experts.

1 Introduction

We consider the problem of reducing road congestions and delays experienced by drivers in a city by letting a wireless sensor and actuators network dynamically control traffic lights. In 2012, according to the Inrix Institute[1], drivers wasted on average 90.3 hours in Brussels, 72.6 hours in Milano, or 63.6 hours in Los Angeles. During the peak hours, users suffered an additional average travel time of 30.2 % in Paris and 27 % in London. Besides travel times, congestion increases pollution and on noise, which makes it a key issue in metropolis. Congestions can have multiple causes (e.g. roadworks, accident, or simple overload) but they tend to expand from their starting point in a first phase. We believe that a reactive and adaptive management of traffic lights could prevent such expansion and help to resorb the situation faster.

Operators often manage each intersection statically: the sequence and durations of the green lights are pre-determined and do not adapt dynamically to the traffic conditions. Detectors can count vehicles on each lane of an intersection, but an operator only uses the data they collect to choose between a few static sequences and timings setups. However, with such data, a dynamic algorithm could take proper local decisions. For example, turning a light green for a blocked direction at an intersection is, at best useless, and can even worsen the problem, as impatient drivers will eventually try to use every available space. In such situations, offloading intersections that are on the border of the congested zone seems more efficient.

In large cities, operators use systems like SCOOT ([11]) or SCATS ([12]) to manage the traffic lights dynamically. These systems are centralized in a control center and require a regular monitoring of the traffic through a network of detectors. These systems have a limited scalability, as they require interconnecting all the detectors through a communication backbone and important computation capabilities. This restricts the deployment of these systems at large intersections and side roads are not considered. The city of London, for example, has about 1600 SCOOT nodes and yet appears as one of the most congested cities according to the Inrix Institute. A decentralized wireless sensor network could manage the capillary network by keeping the data locally.

[1] `http://www.inrix.com/scorecard/`

A wireless sensor network is composed of small and cheap embedded devices that can estimate how many vehicles are present on the different lanes of the intersection, communicate with close nodes, and make simple calculations. The low cost and the ease of installation of these devices, compared to induction loops, permit the creation of a dense network. Once deployed, these devices can exchange information with all the relevant intersections and solve quickly a given situation without involving a central server. The set of relevant intersections can be adapted dynamically to the situation and multihop communication can support data dissemination even when no WAN is available, or when the operator cannot afford wireless data subscriptions for all its devices. In this paper, we present and evaluate a distributed algorithm, TAPIOCA (*distribuTed and AdaPtive IntersectiOns Control Algorithm*) that uses a distributed networks such as a wireless sensor network to acquire data on the vehicles distribution at intersections and to compute and apply a green lights policy.

Section 2 presents classical usage of sensors in ITS and typical deployments at an intersection before looking at related works. We then present a possible hierarchical architecture in Section 3. We then detail, in section 4, a first adaptive traffic lights control algorithm that dynamically selects movements and green light times to reduce the Average Waiting Time (*AWT*) of users without introducing starvation. In Section 5, we present a generalization of this algorithm that allows collaboration and synchronization between close intersections to create *green waves* (sequences of successive green lights). The general algorithm, TAPIOCA, is presented in Section 6. Simulations results obtained through the SUMO simulator are presented in section 7. They compare the AWT and queues lengths achieved by TAPIOCA with state of the art algorithms and with pre-defined plans designed by operational centers.

2 Related Works

2.1 Using wireless sensors in ITS

Sensor nodes in adaptive ITS ([15, 13, 17, 16, 18]) generally feed a queueing model with the number of vehicles present on each lane of an intersection, or with the vehicle arrival process intensity. If these sensors are often radars or induction loops, their cost reserves them to main roads. On smaller roads, magnetometers can record a unique signature for 99 % of the vehicles passing over them ([2]) by measuring the changes on Earth's magnetic field. Corredor *et al.* [4] shows that using these devices can lead to better results than induction loops. They are responsive, easy to install and can be deployed densely, multiplying the number of measurement and action points. Knaian [10] evokes a manufacturing cost lower than $ 30 per unit with a 16-bit micro-controller and a size comparable to a coin. Cameras represent an even cheaper solution, as they do not require roadworks for installation and can achieve a fair accuracy with image processing techniques, even if their angle of vision is limited and sensitive to obstruction. [3] proposes to combine magnetic sensors and cameras in ITS. Both types of sensors can take power out of a battery, or can be connected to the power grid.

Adding a wireless transceiver to these devices allows building a classical WSN. WSNs usually rely on short-range, low cost, low speed, and low power wireless communications. Among the multiple suitable communication standards, IEEE 802.15.4 [8], the Zigbee link-layer technology, provides coverage of about 50-100 meters in the 2.4 GHz band for a maximum data rate of 250 Kbps and a low power consumption [3]. IEEE 802.11p [7] (WAVE – Wireless Access in Vehicular Environments) also represents a viable candidate, as it is expected to be deployed in numerous vehicles and infrastructure devices and operates in the 5.9 GHz DSRC band. 433 MHz technologies are also interesting, as they provide low-throughput and high range communication interfaces. All these technologies share some common characteristics in terms of performance and yield to similar network designs: the communication network formed has all the characteristics of a wireless local area network and may be used as a multihop network, or interconnected through WAN interfaces. The technology is therefore ready for supporting a distributed application like TAPIOCA.

2.2 Sensors deployment

The classical intersection model, used in several academic works, is represented on Figure 1. It is composed of four *directions* (N, E, S, W). In this article, we consider that each direction has two incoming lanes and two outgoing lanes. For an incoming direction, vehicles turning left use the leftmost lane, while the rightmost lane is for vehicles going straight or turning right. We place ourselves in a scenario in which people drive on the right side of the road.

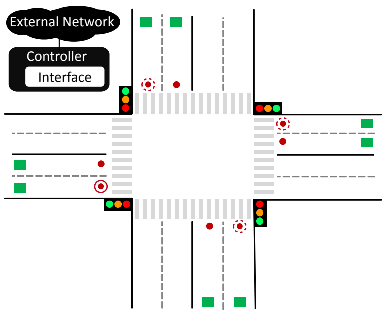


Figure 1: A 4-lanes intersection monitored by 2 sensor nodes per lane

At each intersection, a *controller* defines and enforces a sequence of green and red lights called a *cycle*. A cycle is composed of successive periods called *phases*. In each phase, a subset of the lights is lit green during a certain time, allowing some *movements* to occur simultaneously. Each movement is represented by the cardinal directions of its origin and destination (e.g. *WE*: from West to East). A phase is therefore defined by a set of allowed movements and a duration. We suppose here that a traffic light controls each movement or, at least, each direction.

The traffic on every incoming lane is generally monitored by two sensors: one located close to the traffic light, counting the vehicles departures from the intersection and another one placed at an appropriate distance before the light, counting vehicles arrivals. [13] found that using one sensor is not enough to achieve the best performance: the number of detected vehicles have to be accurate. The distance between these two sensors has an influence on the system performance. [15] propose to set this distance to 8 vehicles. [17] proposes to base this distance on the maximum authorized green time. A single sensor can be used per lane, but results are less accurate. One single sensor can even be positioned per direction, but the vision of the traffic is then limited by the sensors detection range.

However, this architecture is not fully efficient in the multi-intersection case. Figures 1 and 2 shows three different deployments of sensors at a given intersection. These scenarios essentially differ by the number and the positions of the *destination nodes* that monitor and count departures from the intersection. The roles affected to destination nodes will be explained in section 3. In the lane-oriented architecture (figure 1), destination nodes are located on the same lane as arrival nodes. Such a deployment does not distinguish outgoing directions when multiple movements originate on the same lane and also fails to count vehicles blocked in the middle of the intersection. In the direction-oriented architecture (figure 2(a)), destination nodes are located at the entry of the outgoing lanes. Such a deployment better differentiates vehicles exiting the intersection and

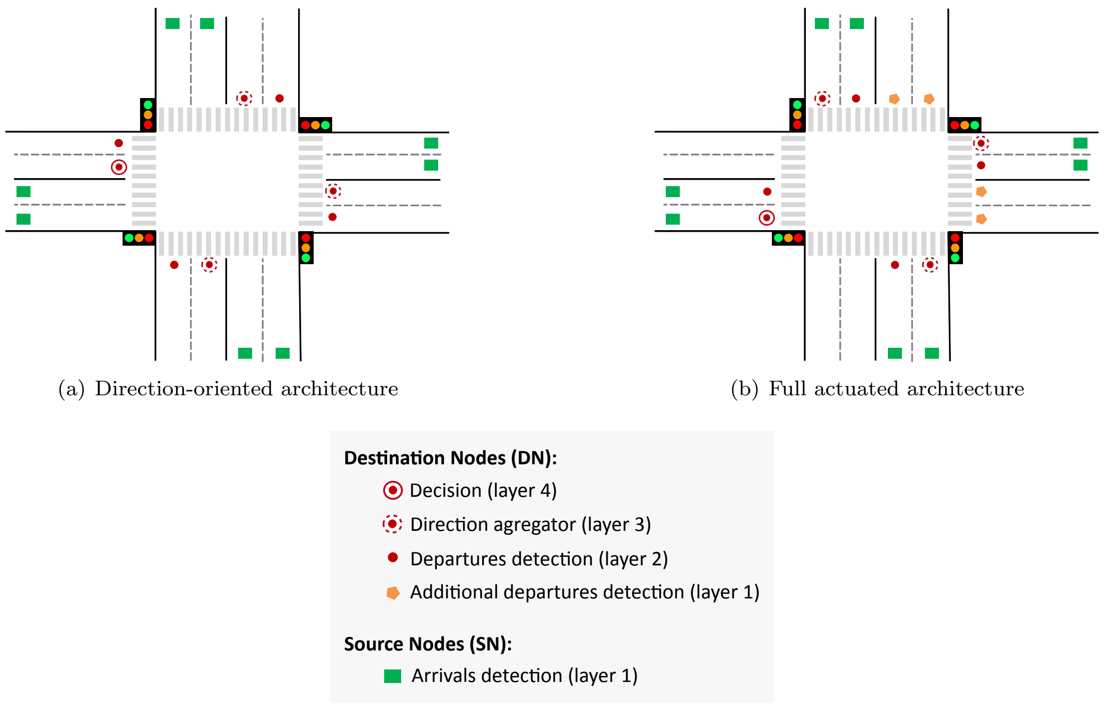

(a) Direction-oriented architecture
(b) Full actuated architecture
Destination Nodes (DN):
Decision (layer 4)
Direction agregator (layer 3)
Departures detection (layer 2)
Additional departures detection (layer 1)
Source Nodes (SN):
Arrivals detection (layer 1)

Figure 2: Two other examples of architectures.

Architecture	Lane-oriented	Direction-oriented	Full actuated
Minimum number of nodes	$Incoming\ lanes \cdot 2$	$Incoming\ lanes + Outgoing\ lanes$	$Incoming\ lanes + Outgoing\ lanes$
Maximum number of nodes	$Incoming\ lanes \cdot 2$	$Incoming\ lanes + Outgoing\ lanes$	$Incoming\ lanes \cdot 2 + Outgoing\ lanes$
Number of communications to count vehicles on a lane	2	$1 + lane\ destinations$	2
Queue length method	Naturally on lanes	With vehicle IDs	Naturally on lanes
Additional gap time when new vehicles come	Yes	Approximately	Yes
Red lights passage detection	Yes	Approximately	Yes
Detection of vehicles blocked on the intersection	No	Approximately	Yes
Identify multiple movements on a single lane	No	Yes	Yes
Finality	Isolated intersections	Multiple small intersections	Multiple intersections

Table 1: Architectures qualitative comparison

allows warning neighbor intersections about the coming flow. However, fully characterizing the vehicle flow requires being capable of recording the electromagnetic signature of each vehicle. The full actuated architecture (figure 2(b)) composes the two previous cases and provides the best detection accuracy. It requires a denser deployment, though. Table 1 compares qualitatively these three scenarios.

2.3 Adaptive traffic lights control systems

Using such a sensors deployment, Yousef *et al.* [15] define the scheduling policy at a single intersection by modeling each movement as an M/M/1 queue. Based on a matrix that identifies conflicting movements, they propose an algorithm that selects the combination of compatible movements that exhibits the highest number of incoming vehicles and computes the green light time proportionally to the total number of vehicles. They then extend their work to a mesh of intersections by selecting first, when the phase begins, the movements that are expected to receive the most vehicles from close intersections. They compute this number by taking into account the time needed to go from one intersection to the other, including stops and slowdowns.

For an isolated intersection with four directions, Tubaishat *et al.* define a phase selection method based on queues sizes in [13]. Zou *et al.* [18] define green light time using fuzzy logic on the vehicle count per minute. None of these three contributions really allows conflicting movements to happen simultaneously. Moreover, they all only take into account the queues size, which may lead to a well-known scheduling problem: starvation.

Zhou *et al.* [17] propose an algorithm that selects the sequence of phases among a set of conflict-free situations according to multiple criteria: the presence of priority vehicles, the duration of the periods when no vehicle is detected, the starvation degree, the total waiting time and the queues lengths. However, this algorithm requires all vehicles to travel at the same speed. [16] extend this work to the multiple intersections case and take into account more parameters. They mix local objectives with the expected traffic flow coming from neighbor intersections. A minimum green time is computed based on the number of vehicles to be treated locally and can be extended depending on incoming flows.

In all these papers, sensors are used as simple detectors that report measurements to a central entity. However, such an organization does not scale and has a low fault tolerance: the controller is a single point of failure. There are multiple possibilities to organize the network formed by the sensors and to distribute the computation and storage duties over this local network. Not only does it improve fault tolerance, but also communication and decision latency, as data is processed and decisions are taken close to the information collection point and to the lights controller.

3 TAPIOCA: network organization

TAPIOCA can run on any of the architectures represented on Figures 1 and 2. By default, TAPIOCA distributes tasks among the different sensors present at an intersection. The *destination nodes* (DN) nodes measure departures when the light is green. They are typically located at the traffic light on each input lane or at the entrance of each output lane (case of Figures 2(a) and 2(b)). The *source nodes* (SN) measure arrivals continuously. They can be located at a fixed distance of the light, to consider as soon as possible the entrances on the intersection (but not too far away, to avoid errors due to lane changes). The ideal distance of SN depends on the traffic conditions, therefore these nodes could also be mobile (e.g. mounted on rails) or could be dynamically selected among a large set of deployed sensors.

According to the literature, the distance between SN and DN nodes should correspond to the number of vehicles that can cross the intersection when the traffic light green duration is at its maximum allowed (denoted by T_{max}). T_{max} is generally bounded to limit waiting time badly perceived by users. This distance should be set to $N * L_{veh}$, where N denotes the number of vehicles able to pass in T_{max} seconds and L_{veh} the average length of a vehicle. If T_s denotes the startup time of a vehicle and if T_H represents the headway time that separates two vehicles, $N = \frac{T_{max}-T_s}{T_H}$. According to [14], $L_{veh} = 6$, $T_s = 4\,s$ and $T_H = 2\,s$ are realistic values and can be adjusted empirically. Based on previous results ([5, 6]) and [14], a typical value for this distance would be 75 meters, which corresponds to a T_{max} value of 30 s.

Among the DN nodes, one *direction aggregator* is elected per direction to collect, process and aggregate the traffic of all DN nodes located lanes emmiting in the same direction. These aggregators then report to a single decision point, possibly elected among the sensors, which is in charge of computing the schedule and communicating it to the lights controller.

This organization, which not mandatory for TAPIOCA, presents certain advantages. First, direction aggregators possess the information on the total number of vehicles going out of the intersection in a given destination. They are able to send this information to all the next intersections and are good candidates to act as gateways towards these neighboring intersections if the network functions in a multihop manner. Second, all the aggregators roles are exchangeable. This means that a distributed election protocol can be in charge of selecting the best candidates. Such election algorithms have a reasonable cost ([9] operates in $\log(n)$ time) and can be based on nodes capabilities (memory, computation power, etc.). Finally, these roles can be passed over between sensors when failures happen and this architecture provides a certain level of fault-tolerance.

4 The isolated intersection case

In [5], we proposed a WSN architecture and an algorithm for controlling green lights on a single intersection. The sensor nodes deployed at each intersection exchange information using wireless communications and agree on the intersection schedule. Instead of defining cycles, the algorithm works at the phase granularity, selecting dynamically movements with two objectives in mind: reducing the average waiting time and avoiding starvation. It allows lightly conflicting movements to happen simultaneously. Such movements are movements that do not impact users safety, i.e. when obvious priority rule between the movements exist. This algorithm yields to a reduced average waiting time. As this algorithm is the basis of the multiple intersections control algorithm presented below, this section presents an improved version that follows the following principles: movements are first classified by evaluating and weighting (Sec. 4.1) multiple objectives. Then, a phase is created by associating movements together, taking into account potential conflicts with other lanes or with pedestrians (Sec. 4.2, 4.3 and 4.4). Finally, the green lights duration is calculated in function of the traffic (Sec. 4.5).

4.1 Ranking movements based on two objectives

Let us consider a possible movement (s, d) going from direction s to direction $d \in D$, D denoting the set of all possible directions (N, W, S, E in a 4 directions intersection). The decision node (layer 4) knows the whole vehicles distribution at the intersection. It is thus able to associate a local score $S(s, d)$ to the movement (s, d). This score depends on the number of vehicles present on the incoming lanes that compose the movement ($N^{(s,d)}$) and on the time since the movement was last selected ($T_F^{(s,d)}$), i.e. since it last had a green light.

We need to combine the metrics reflecting both objectives (load and interval between successive selections) in a single expression to define formally this score. To this extent, we normalize both objectives using a generic function, $\gamma(o^{(s,d)})$, whose aim is to bring an objective $o \in \{T_F, N\}$ in the $[0;1]$ and to make it dimension-free so that they can be compared. If $o^{(s,d)}$ denotes one objective value for the movement (s,d), $\gamma(o^{(s,d)})$ is defined naturally as the ratio of the movement objective value over the cumulated objective value of all movements:

$$\gamma(o^{(s,d)}) = \frac{o^{(s,d)}}{\sum\limits_{\{a,b\} \in D} o^{(a,b)}}.$$

Once normalization is realized, the classical approach would be to define the score as a linear combination of $\gamma(T_F^{(s,d)})$ and $\gamma(N^{(s,d)})$. However, as we seek a ranking rather than a notation, we would like to give more weight to the movements that have a number of vehicles significantly higher than the other ones, or that have not been selected since a long time. That's why we define the score as a weighted sum of the squares of the normalized metrics:

$$S(s,d) = W_N \cdot \left(\gamma(N^{(s,d)})\right)^2 + W_{T_F} \cdot \left(\gamma(T_F^{(s,d)})\right)^2,$$

where W_N and W_{T_F} are user-defined weights, which can be defined and changed by operators empirically to favor performance (AWT) or users experience (starvation). By default, these weights are identical.

Squaring the sub-objectives metrics keeps the values in the same interval but emphasizes differences more and more as the values increase. This latest property has the desired effect: if one of the sub-objectives has a high value for a given movement compared to the other ones, the corresponding contribution to the score will be significant enough to favor its selection.

If no vehicle is present on the lanes originating at direction s, we force the score to be null for all the relevant movements, even if the movement has not been selected for long: $\forall d \in D, S(s,d) = 0$.

4.2 Identifying conflicting movements

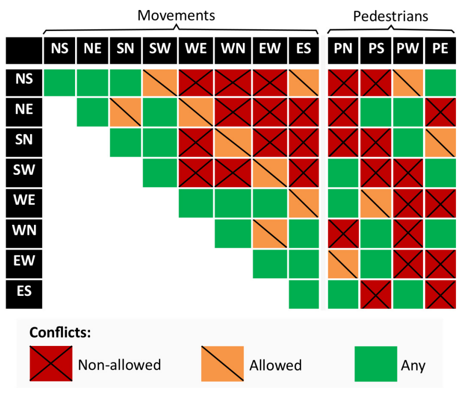
4.2.1 Based on a conflict matrix

once scores have been computed for each movement, the decision node examines which movements can be combined in order to create a new phase. To this extent, it uses a conflict matrix, which indicates the movements that can safely be performed simultaneously. Figure 3 represents one such conflict matrix with three levels: green elements indicate non-conflicting movements and red elements indicate that movements cannot happen together, e.g. for safety reasons. In between, orange elements indicate conflicting movements with no real safety issue. If such cases, which generally involve left-turn movements, do not pose safety issue, the conflict can however limit the number of vehicles that can cross the intersection and potentially introduce inter-blocking situations. That's why such movements should ideally receive green light simultaneously only when the number of vehicle is low. Section 4.3 describes how TAPIOCA allows to handle such situations. This matrix also includes columns for the pedestrian crosswalks.

This matrix is utilized to combine individual movements that could compose a valid phase by summing their scores. The selected phase is the one that achieves the highest score.

4.2.2 Based on a static phase

finding the best combination of movements when the conflict matrix allows multiple simultaneous movements requires calculation. In our simulations – based on hundreds or some times thousands



	Movements								Pedestrians			
	NS	NE	SN	SW	WE	WN	EW	ES	PN	PS	PW	PE
NS	🟩	🟩	🟩	🟧	🟥	🟥	🟥	🟧	🟥	🟥	🟧	🟩
NE		🟩	🟧	🟩	🟧	🟥	🟥	🟥	🟥	🟩	🟩	🟥
SN			🟩	🟩	🟥	🟧	🟥	🟥	🟥	🟥	🟩	🟧
SW				🟩	🟥	🟥	🟧	🟥	🟩	🟥	🟥	🟩
WE					🟩	🟩	🟩	🟧	🟩	🟧	🟥	🟥
WN						🟩	🟧	🟩	🟥	🟩	🟥	🟩
EW							🟩	🟩	🟧	🟩	🟥	🟥
ES								🟩	🟩	🟥	🟩	🟥

Conflicts:

🟥 Non-allowed 🟧 Allowed 🟩 Any

Figure 3: Example of a conflict matrix

intersections (see 4.6, 7.1 and 7) – we propose an alternate solution that consists in defining the list of possible phases based on the pre-determined light plans, which are realistic and optimized. We suppose that, in the same way as the SUMO simulator ([1]), each light plan phase defines which lanes have the green light. Based on each phase, we can easily retrieve each movement (s, d) associated to the lanes having the green light and compute the score $S(s, d)$. The selected phase is the phase that achieve the highest score summation.

4.3 Letting lightly conflicting movements occur together

Vehicles turning left usually have the lowest priority, i.e. they are blocked whenever a vehicle coming from the movement in the opposite direction passes. The opportunity to let the two movements happen simultaneously therefore depends on the load of both movements.

Classical strategies consist either in letting each of these movements occur in a dedicated phase, or to separate the phase into two sub-phases: the first sub-phase allowing concurrent movements and the second one letting only the low priority movement happen. Such compound phase is slightly different from the two successive phases case, as in introduces a dependency: the second sub-phase should happen immediately after the first one. TAPIOCA adopts this second solution and allows to create a sequence of phases.

4.4 Managing pedestrian crosswalks

Pedestrians crosswalks can easily be included into TAPIOCA's model by adding entries to the conflict matrix, as shown on Fig. 3. Crosswalks could be considered as classical vehicles lanes, with their dependencies and included in the conflict matrix per se. However, they behave slightly differently, as it is more difficult to count the number of awaiting "vehicles" on these lanes. TAPIOCA adopts an alternate approach: crosswalks do not participate directly into the phase selection algorithm, but once a phase composition is determined, all the crosswalks that are not in conflict with any selected movement are selected as well. This rule can be lightened,a s we could tolerate moderate conflicts (e.g. when vehicles turn right and have to give passage to the pedestrians).

Considering pedestrian can be done very easily, by integrating pedestrian crossings in the conflict matrix, as the example on Figure 3 show. When the phase is know, the pedestrian lights that do not conflict with selected movements are added to the phase. In the same manner as the movements, one pedestrian crossing may has one or more allowed conflict: in this case, some vehicles can interfere with pedestrian crossings, but they are not considered dangerous or charged, they are not prioritized. The conflict matrix must be configured to allow all crosswalks have the green light. In this case, pedestrian crossings are not an obstacle to the phase composition: the traffic lights are not set aside.

In special cases, no conflict with pedestrians is desired or it is impossible to configure a matrix conflict with any conflict between the vehicle movements and the pedestrian crossings. In this case, a pedestrian crossing P_n can has the same proprieties than a vehicle movement: $T_F^{P_n}$ is the last time the pedestrian light was green and N^{P_n} is a pedestrian presence indicator, which can be computed accurately with cameras, piezoelectric sensors or more simply with pushbuttons.

In reality, light plans are defined with traffic lights but also with pedestrian lights. Therefore, use a conflict matrix to manage pedestrian crossing is not mandatory: it can use the original pre-determined light plan (see Section 4.2.2).

4.5 Defining a phase duration

The lifetime of an intersection is, in TAPIOCA, a succession of phases. Unlike most strategies, there is no explicit notion of cycle, the phases succeed and TAPIOCA ensures that all movements are selected regularly. When a phase ends, the lights are set orange during a time T_{orange}, traditionally fixed to 3 s. Then all light turn red during a guard time T_{secure}. This leaves enough time to the controller to determine which movements will compose the upcoming phase. Once movements are selected with respect to the various criteria exposed in the previous paragraphs, the controller compute the phase's effective green time: T_p.

Let us denote by T_s the start-up time of the first vehicle in line, which is fixed to 4 s according to [14]. T_h represents the headway time that separates two vehicles passages. It is fixed to 4 s according to [14]. T_{max} represents a phase maximal time, which is either defined by the user, or determined in order to balance performance and users experience aspects, as discussed in section 3. T_p is then computed in function of the load on the largest selected incoming lane, as if the intersection intended on emptying the lanes, as follows;

$$T_p = min(T_s + N_{max} \cdot T_h,\ T_{max})\,,$$

where N_{max} is the number of vehicles on the most populated lane that will be granted green light in the phase. This time is bounded by T_{max} to avoid capturing the intersection. In the case of $T_p < T_{max}$, additional vehicles can arrive on these lanes. In this case, we choose to let the green time increase by T_h, until no new arrival happens, or until $T_p = T_{max}$.

The value of T_{max} has an influence on the overall performance. In [5], we found that the optimal T_{max} value for an intersection of Amiens (France) lied between 25 s and 35 s. The proper T_{max} is smaller when conflicts are forbidden, as letting all movements happen in this situation requires more phases, leaving less time to a single phase. In these first simulations, we chose to set empirically T_{max}, by testing a set of values from 15 s to 70 s and using the best result.

If such an empirical approach is always possible, we propose here to adjust T_{max} by taking into account the cycle length (T_c) defined by static light plans when available.

T_c is set to limit the time between two repetition of the same phase and is defined depending on intersection traffic and configuration. On an arbitrary intersection, we set T_{max} proportionally to the number of vehicles composing the phase:

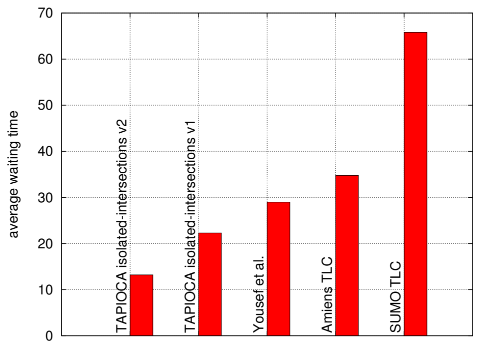
$$T_{max} = \frac{N_{max} \cdot T_c}{\sum\limits_{\{a,b\} \in D} N_{max}^{(a,b)}}.$$

Thus, we are assured to get a coherent T_{max} value with respect to the diversity of intersections. This method is applied in our simulations, and proves itself useful and efficient for large scenarios such as TAPASCologne (Section 7).

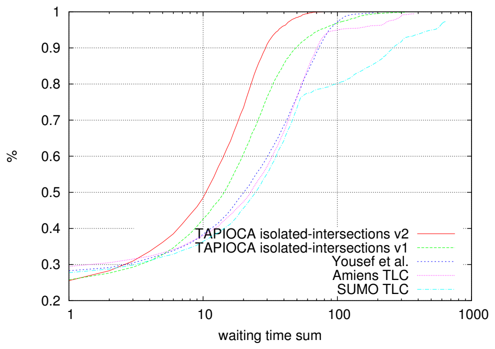
4.6 Evaluation based on Amiens isolated intersection

The simulations presented on figure 4 are performed using SUMO ([1]) and are based on one intersection of Amiens city (France), for which we have at our disposal the real traffic data, the vehicle distribution and the figures of the static policy (cycles, phases and timings). Each simulation lasts 7,200 program steps, which represents 7,200 s. Results presented on figure 4 are the average values calculated on the 4,381 vehicles that traverse the intersection during this simulation time.

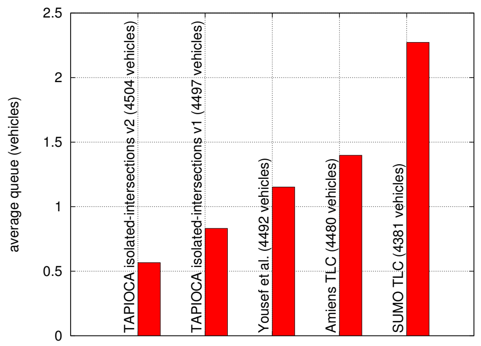
Figures 4(a) and 4(b) show that the improved version of TAPIOCA achieves the best average waiting time compared to its previous version (40% reduction), to an adaptive algorithm from the literature (55%), to the static light plan from Amiens city (62%) and to the SUMO generated light plan (80%). Figure 4(c) shows that TAPIOCA also produces the shortest queues on average.



(a) Average waiting time.



(b) CDF of the waiting time (log scale).



(c) Average queue length.

Figure 4: Simulations on Amiens isolated intersection.

5 TAPIOCA : the multiple intersections case

In this section we propose a generalization of the previous algorithm to multiple adjacent intersections, by defining a method to synchronize close intersections and to create green waves. We assume that each intersection can communicate with its adjacent intersections through the WSN. It may use additional relays, forming a multihop network, or a WAN connection if available when the distance between two intersections is larger than the nodes transmission range. We also assume that the clocks of nodes are synchronized with sufficient accuracy for the application (i.e. approximatively one second).

In [6], we proposed a first approach to manage the multiple intersections case, by selecting, similarly, the movements composing the upcoming phase based on local and distant measurements combined in a global score, that mixed three objectives: first, a local score was computed with the method described in section 4.1. Then, scores were exchanged between close intersections so that an intersection favored movements that had the highest overload capacity (i.e. avoid sending too many vehicles to already congested areas) and that tried to synchronize successive intersections to create green waves. This first algorithm will be designated by “TAPIOCA multiple intersections v1” in the simulations.

Green waves are paths of successively green lights, synchronized so that vehicles do not slow down. It is among the most efficient techniques to discharge a network, as it makes traffic more fluid. The usual implementation of green waves supposes that vehicles travel at the limit speed and synchronization needs to be adapted when congestion appears. TAPIOCA multiple intersections v1 did not take into account the load and its performance was not optimal when congestion level significantly delayed vehicles between two neighbor intersections.

In the version of TAPIOCA introduced in the present article, we chose to modify the philosophy behind intersections synchronization. Instead of directly taking into account information coming from neighbors in the calculation of a complex score, we let each intersection take its own decisions locally, as if it were isolated. The result is a set of active movements and an expected number of vehicles that will go, during the next phase, from the originating intersection (I_1) to some of its neighbors.

Let us denote by I_2 one of these neighbors that should expect to receive an incoming flow soon. I_1 could send a message to "warn" I_2 about the incoming flow. However, this would generate a high data traffic, increasing the congestion and loss probability in the communication network, especially when using a limited bandwidth radio communication technology. That’s why we choose to limit the transmission of these messages to the only case when I_2 is less loaded (i.e. has a smaller total number of vehicles) than I_1. The intuition behind this filtering is a pure greedy strategy: I_2 should only consider interrupting its own cycle if it helps reduce the load of a more loaded neighbor intersection. Otherwise, it should continue with its own local schedule.

This synchronization message contains the number of vehicles I_1 expects to send to I_2 and an estimation of the time required by these vehicles to reach I_2. This time can be evaluated simply based on the distance between the two intersections, or by exchanging a sample of the magnetic signatures of the individual vehicles between I_1 DN nodes and I_2 SN nodes, and by taking into account the headway (T_H), orange (T_{orange}), secure (T_{secure}) and startup (T_s) times (see Sec. 3).

When I_2 receives such a message, it evaluates locally the opportunity to break its ongoing cycle to favor a green wave. Indeed, the message filtering that was performed by I_1 necessarily relies on slightly outdated information. It compares both expected gains, i.e. the number of vehicles that will cross the intersection if the current phase is maintained and if the green wave interrupts the current phase. If the choice is in favor of the green wave, I_2 determines a new phase composition, applying the classical TAPIOCA algorithm with the additional constraint to necessarily select movements that correspond to the incoming flow.

Synchronization messages arrive asynchronously at an intersection and one intersection could have to deal with multiple requests during the same phase. In this case, it selects the strategy that maximizes the expected number of vehicles that will cross the intersection during the next phase.

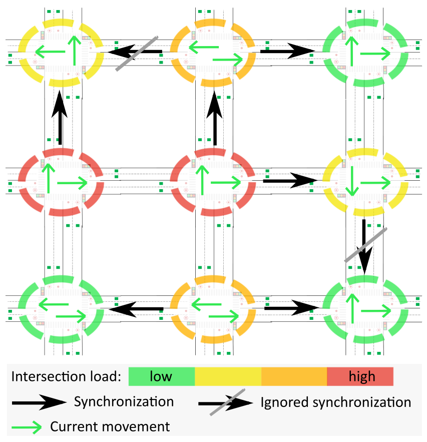


Figure 5: Synchronization messages sent between intersections and filtered by the destination in a small scenario

Figure 5 represents a small network of 9 intersections. Each of these intersections has an ongoing phase, composed by the movements materialized by the inner green arrows. The synchronization messages that are effectively sent are represented by black arrows between intersections. Note that an intersection neither sends messages to neighbors that are not the destination of the current phase movements, nor to more loaded neighbors. Once these messages are received, the destination intersections take decisions and ignore requests when the gain from local objectives is higher than the gain from the green waves. Such ignored requests are represented by striked arrows.

6 Under the Hood: TAPIOCA algorithm and communication flows

This section details the full *TAPIOCA* algorithm, which is distributed over the intersections and gives additional precisions. TAPIOCA's aim is to select the movements composing phase $P+1$ at the end of phase P, based on the principles exposed in sections 4 and 5). This version of TAPIOCA is meant to be implemented on the architecture 1 or 2(b), i.e. with departure detectors on the same lanes as the arrival detectors. The case of architecture 2(a) is detailed in [6]. TAPIOCA is decomposed in 7 steps:

1. **Counting the vehicles on each lane (SN and DN nodes)**: For each lane i, at the end of phase P, every SN node (layer 1) sends $N_i^A(P)$, the count of arrivals that occurred during phase P, to its corresponding DN node (layer 2). In parallel, each DN node has monitored the number of departures during the phase ($N_i^D(P)$). Each DN node updates $N_i^P(P+1)$, the number of vehicles that are present on the lane at the beginning of phase

P+1 according to the following formula:

$$N_i^{P+1} = N_i^P + N_i^A(P) - N_i^D(P)\,.$$

It then transmits all N_i^{P+1} values to the direction aggregator (layer 3) node that manages the incoming direction DN belongs to.

2. **Per-direction aggregation (direction aggregator nodes)**: For each direction y, the aggregator maintains the time elapsed since the last selection of the movements starting at y, T_F^y, to detect and prevent starvation. For each movement starting at y, it sums the N_i^{P+1} values received from every relevant lane to get $N^{(a,b)}$, the total queue length for the movement (a, b).

   If one lane i is the origin of M movements ($M > 1$), we choose to set by default for each movement (a, b), $N^{(a,b)} = \frac{N_i^{P+1}}{M}$, to avoid counting M times the vehicles of the lane i. If additional sensors are installed on the output lanes, they can cooperate with the DN node of the lane having multiple movements to determine a coefficient for each movement. For example, if an average of 60% vehicles of the lane i follow the movement (a, b), then we can set $N^{(a,b)} = \frac{N_i^{P+1}}{0.6}$.

   Finally, the direction leader transmits these two values to the network leader (layer 4) sensor. Aggregators, which also manage the synchronization messages received from neighbor intersections, transmit at the same time their messages to the decision node.

3. **Next phase composition (decision node):** once it has received data from all aggregators, the decision node computes the local scores $S(s, d)$ of the different movements (Section 4.1). It combines movements using the conflict matrix or a pre-determined set of phases from the static light plan and select the combination that receives the best score. At this stage, additional criteria can be considered (e.g. emergency vehicle detection, combination avoiding in case of accident detection).

4. **Next phase duration (decision node):** once the movements have been selected, the green light time is set according to Section 4.5.

5. **Transmission to neighbor intersections (decision node, direction aggregators):** once the decision node has defined the next phase composition and timing, it transmits this information to the aggregator nodes, alongside with a possible intersection synchronization message and the number of vehicles on the intersection. Each aggregator node then forwards this information to the neighbor intersections.

6. **Phase application (decision node):** the decision node instructs the controller to turns the specified lights on for the specified time. This marks the beginning of phase $P + 1$.

7. **During the phase $P + 1$ (decision node)**. If a synchronization message arrives and is considered relevant, the decision node relaunches the phase selection process, forcing the selection of the movements involved in the green wave. When the phase finishes, then the decision node applies the left-turn gap process (Section 4.3) if required.

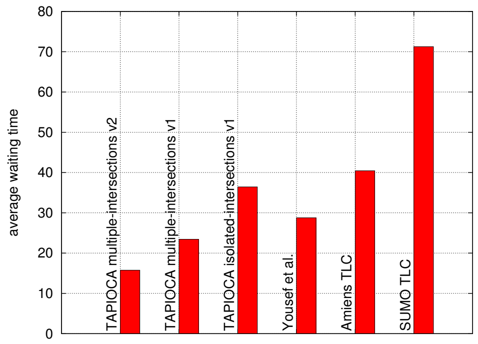
8. **Inter-intersection vehicles monitoring (DN nodes):** during phase $P + 1$, the DN nodes of selected directions may send the vehicles timed signatures to the corresponding DN nodes of the next intersections that should see these vehicles pass. This allows estimating the time that is required to go from one intersection to the other. This delay can be used to tune the green waves synchronization process, and its variations allow an early detection of upcoming congestion.

This algorithm has been specified to allow easy improvement, taking for example into account failure of a sensor or information coming from a vehicular network, or from cell phones or connected GPS devices.

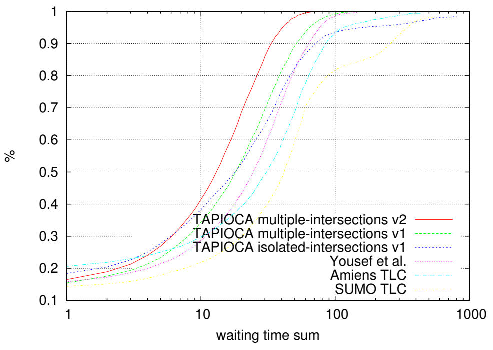
7 Simulations

We evaluated TAPIOCA using the SUMO 0.17 (Simulation of Urban MObility, [1]), open-source, discrete time, continuous space and microscopic traffic flow simulator. Our simulations were performed over more than *1,425,137* cumulated trips distributed into *32* distinct scenarios: the complete results, and more figures, are available online[1]. We detail in this section representative metrics and scenarios.

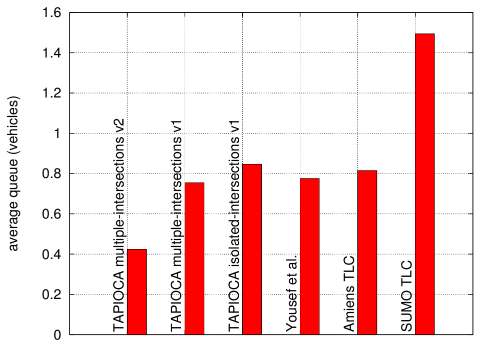
7.1 Scenario 1: Amiens multiple intersections



(a) Average waiting time.



(b) CDF of the waiting time (log scale).



(c) Average queue length.

Figure 6: Performance comparison: Amiens multiple intersections scenario.

The simulations presented on figures 6, similarly as Section 4.6, are based on three intersections of Amiens city (France), for which we have at our disposal the real traffic data, vehicle distribution and the true policy (cycles, phases, synchronizations and timings). Each simulation ran during 7,200 program steps, which represents 7,200 s, and is traversed by 5,538 vehicles.

[1]`http://tapioca.sfaye.com/`

Figures 6(a) and 6(b) show the comparison between this version of TAPIOCA, TAPIOCA multiple intersections v1 ([6]), TAPIOCA isolated intersection ([5]), the algorithm of Yousef *et al.* ([15]), the Amiens real light plan and the default SUMO traffic lights policy.

Simulation results show that TAPIOCA achieves the best average waiting time. It is 32% better than TAPIOCA multiple intersections v1, 56% better than TAPIOCA isolated intersection, 45% better than Yousef *et al.*'s algorithm, 61% better than the Amiens light plan and 78% better than SUMO default policy. Figure 6(c) shows that TAPIOCA also generates the shortest average queues.

7.2 Scenario 2: TAPASCologne

TAPASCologne[2] is one of the largest – if not the largest – freely available traffic simulation data set for SUMO. It models the city of Cologne (Köln, Germany) based on the OpenStreetMap cartography and two hours of traffic, i.e. 71,151 vehicles.

The results are presented on Figure 7 and show the average waiting time for the 71,151 vehicles alongside with the average queue length. We excluded previous TAPIOCA versions, as the results were not different from the Amiens case. We can see, on Figures 7(a), 7(b) and 7(c), that TAPIOCA still achieves the best waiting time, compared to Yousef *et al.*'s algorithm (36% reduction) and to the pre-determined light plans of TAPASCologne / SUMO (48% reduction). Figure 7(d) shows that TAPIOCA achieves also the lowest queue, on average. Finally, figures 7(e) and 7(f) show the average time between two successive selections of the same movement: we can see that TAPIOCA achieves also the lowest time, on average.

7.3 Scenario 3: grid networks

Figure 8 shows the performance of TAPIOCA on a square grid of 64×64 intersections (4,096 intersections). Each simulation lasts 7,200 s and sees 48,598 vehicles pass on average, with an arrival process intensity of $\lambda = 8$ vehicles per second. Other simulations realized on 16, 64, 256, 1,024 and 4,096 intersections, with an intensity $\lambda = 0.5$, 1, 2, 4 and 8 are available online[1].

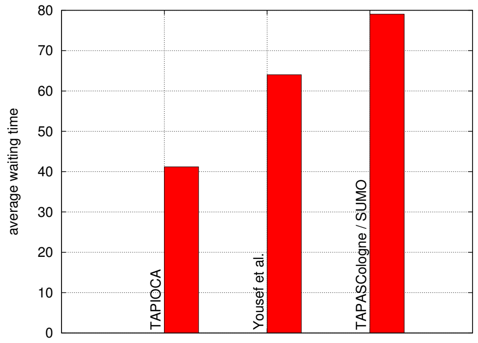
Here, we can see on Figures 8(a), 8(b) and 8(c) a consequent difference between our algorithm and SUMO pre-determined light plans: our algorithm create green waves and reduces the average waiting time by 97%. Compared to Yousef *et al.*, our algorithm reduces the average waiting time by 85%.

8 Conclusion and future works

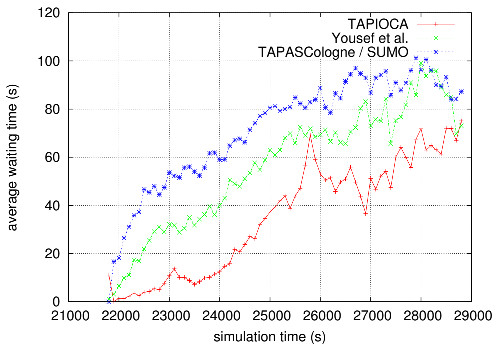
In this paper, we propose and evaluate a distributed adaptive traffic lights control algorithm for multiple intersections that uses a WSN. Based on three different scenarios, we show that this algorithm is able to achieve a better waiting time, travel time and queue length than a predetermined solution, but also than adaptive solutions. Besides the raw performance of the algorithm, these results show that there is an interest in managing the traffic at the intersection granularity. The distributed architecture allows to react quickly to the congestion situations by taking local decisions.

We are currently working on evaluating the communication network performance by using co-simulation between SUMO and OMNeT++. This evaluation shall allow us to test the performance of TAPIOCA under realistic radio conditions, i.e. when channel losses happen, or when communication load increases. This study shall also allow us to determine a fault tolerance

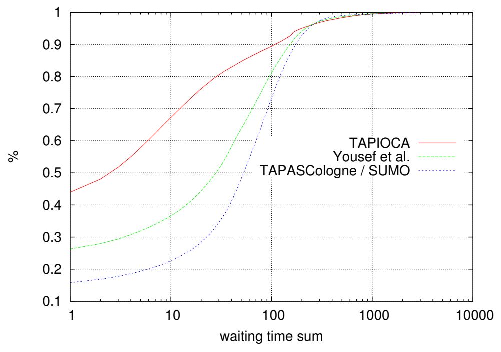
[2]`http://sumo.sourceforge.net/doc/current/docs/userdoc/Data/Scenarios/TAPASCologne.html`



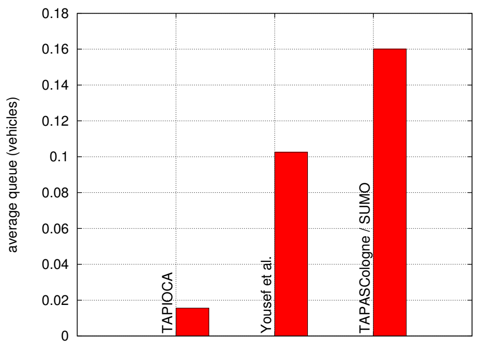
(a) Average waiting time.



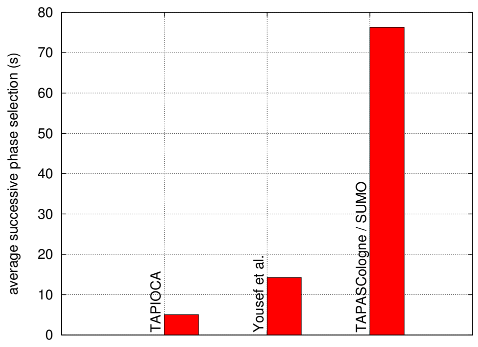
(b) Waiting time evolution.



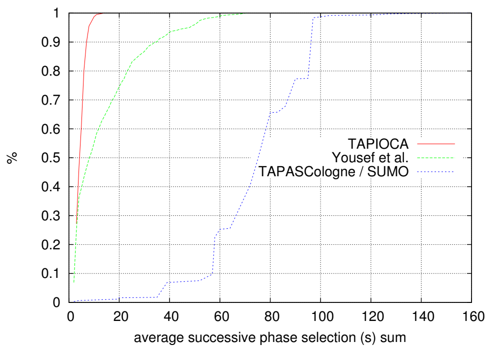
(c) CDF of the waiting time (log scale).



(d) Average queue length.

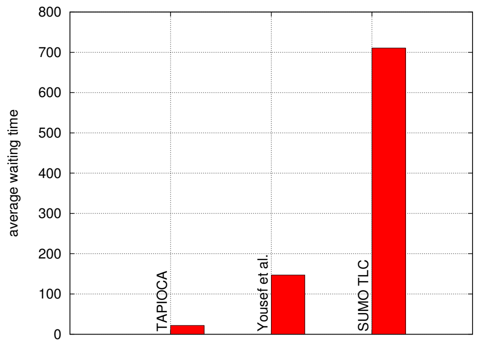


(e) Average movement selection interval.

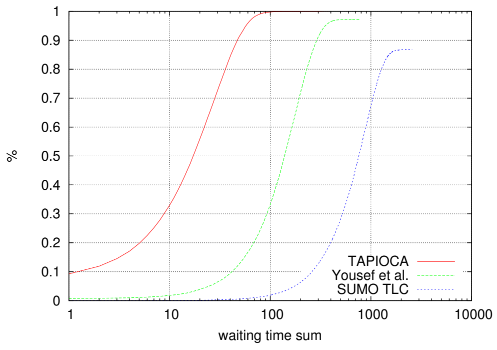


(f) CDF of the movement selection interval.

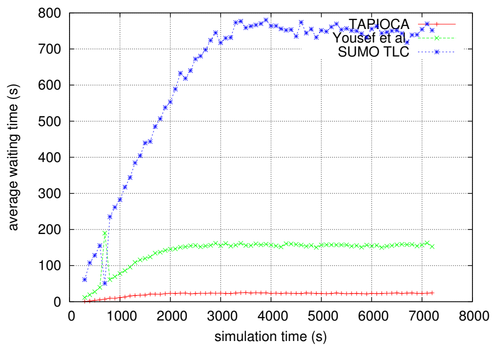
Figure 7: Performance comparison: TAPASColgogne scenario.



(a) Average waiting time.



(b) CDF of the waiting time (log scale).



(c) Waiting time evolution.

Figure 8: Performance comparison: Grid scenario.

strategy and discuss about security issues, when the vehicles are not correctly identified by magnetometers or when there is no information coming from sensors or from an adjacent intersection. In this second version of TAPIOCA, we also limited the communication between intersections to direct neighbors. However, we could benefit from information coming from a greater distance, especially when building green waves. The question of the appropriate communication distance remains open, and its answer will probably depend on the considered scenario, which would plead in favor of a dynamic approach, the communication distance being set according to the current transportation system congestion level. At a longer term, we also plan on integrating a vehicular network or cell phones as additional sources of data.

Acknowledgments

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

References

[1] 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)*, pages 63–68, Barcelona, Spain, Oct. 2011.

[2] 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*, 1917(-1):173–181, Dec. 2005.

[3] M. Collotta, G. Pau, V. Salerno, and G. Scatá. Wireless sensor networks to improve road monitoring. *InTechOpen*, 2012.

[4] 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, June 2008.

[5] S. Faye, C. Chaudet, and I. Demeure. A distributed algorithm for adaptive traffic lights control. In *15th IEEE Intelligent Transportation Systems Conference (ITSC 2012)*, Anchorage, USA, Sept. 2012.

[6] S. Faye, C. Chaudet, and I. Demeure. A distributed algorithm for multiple intersections adaptive traffic lights control using a wireless sensor networks. In *UrbaNe Workshop (held in conjunction with ACM CoNEXT 2012)*, pages 13–18, Nice, France, Dec. 2012.

[7] IEEE Standard for Information technology. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 6: Wireless Access in Vehicular Environments, 2010.

[8] IEEE Standard for Local and metropolitan area networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), 2011.

[9] T. Jurdziński, M. Kutyłowski, and J. Zatopiański. Efficient algorithms for leader election in radio networks. In *21th annual symposium on Principles of distributed computing*, pages 51–57. ACM, 2002.

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

[11] D. Robertson and R. Bretherton. Optimizing networks of traffic signals in real time-the scoot method. *IEEE Transactions on Vehicular Technology*, 40(1):11 –15, Feb. 1991.

[12] A. Sims and K. Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on Vehicular Technology*, 29(2):130 – 137, May 1980.

[13] 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.

[14] US Dept. of Transportation, Federal Highway Administration, Office of Operations. *Traffic control systems handbook*. 2005. http://ops.fhwa.dot.gov/publications/fhwahop06006/.

[15] 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*, 26(3), May 2010.

[16] B. Zhou, J. Cao, and H. Wu. Adaptive traffic light control of multiple intersections in wsn-based its. In *73rd IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2011.

[17] 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, Sept. 2010.

[18] 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.

Dépôt légal : 2013 – 3e trimestre
Imprimé à Télécom ParisTech – Paris
ISSN 0751-1345 ENST D (Paris) (France 1983-9999)

Télécom ParisTech

Institut TELECOM - membre de ParisTech

46, rue Barrault - 75634 Paris Cedex 13 - Tél. + 33 (0)1 45 81 77 77 - www.telecom-paristech.frfr

Département INFRES