Hybrid Petri net model of a traffic intersection in a urban network

C. Renato Vázquez, Herman Y. Sutarto, René Boel, Manuel Silva

***Abstract*— Control in urban traffic networks constitutes an important and challenging research topic nowadays. In the literature, a lot of work can be found devoted to improving the performance of the traffic flow in such systems, by means of controlling the red-to-green switching times of traffic signals. Different techniques have been proposed and commercially implemented, ranging from heuristic methods to model-based optimization. However, given the complexity of the dynamics and the scale of urban traffic networks, there is still a lot of scope for improvement. In this work, a new hybrid model for the traffic behavior at an intersection is introduced. It captures important aspects of the flow dynamics in urban networks. It is shown how this model can be used in order to obtain control strategies that improve the flow of traffic at intersections, leading to the future possibility of controlling several connected intersections in a distributed way.**

I. Introduction

Coordinated control of traffic lights in an urban area offers good prospects for reducing travel time and pollution due to traffic, especially under rush hour conditions. The control action that can be used in order to improve the behavior of the system consists in selecting the red-to-green switching times of traffic signals. Various approaches to the design of control laws for switching these traffic lights have been proposed, and in many cases also implemented, from the 1950s onwards. Nevertheless, given the combinatorial nature of the coordination problem for a network, and given the complicated dynamics of the system due to the huge number of vehicles that inhabit a network under rush hour conditions, there is still a lot of scope for improving the controller agents in such a network of traffic lights. A lot of work is currently going on in the field (see e.g. [1] for a recent contribution).

The issue is to find a good compromise between locally minimizing the delays of vehicle streams at each intersection, while maintaining global stability of the network. These two goals may be contradictory since a local optimization may actually starve the inflow of traffic in some downstream intersections during some intervals of time [2]. Good coordination thus requires a model that allows the efficient representation of both the delays at intersections, and the flows of traffic between intersections. In this sense, this paper discusses a new methodology for representing the dynamic behavior at intersections in a urban traffic network, allowing to capture the key information about incoming flow from neighboring intersections (in an abstracted way).

A lot of heuristic optimization algorithms have been used in order to generate efficient control strategies for traffic light control. Neural networks, box optimization methods, ant colony optimization, and others have been used. Some model based optimization tools were actually implemented commercially, such as the CRONOS tool [3] from INRETS. Over the last few years a number of authors have considered various formal model based optimization approaches to this problem. Interesting works are those of Porche and Lafortune [4]. The main difficulty is the size of the optimization problem, which forces the use of macroscopic models and of distributed control and optimization techniques [5], such as distributed model predictive control [6].

The basic modeling tool used in this work is timed Petri nets. A paper that is closely related to ours is [7] where the heavily populated discrete timed Petri net is aggregated into sets of piecewise linear equations. This paper is different in that we try to relax the heavily populated discrete event model, while keeping the Petri net structure intact as much as possible. Given that the purpose of the control actions is to improve the performance of the system under heavily loaded conditions it is natural that we use a fluidification of some of the transitions in these timed Petri net models. For this example, this means that we do not look at each individual vehicle movement at an intersection, nor do we model the detailed trajectory of each vehicle along the links connecting intersections. The model only describes the flows of vehicles through the intersections and links. The overall model proposed here is hybrid, including both fluidified transitions and discrete transitions, since behavior of the traffic lights themselves, and some queueing networks at traffic sensors, are represented by timed, discrete Petri nets. Fast simulations of this model have been implemented. This simulation tool has been used to search for optimal red-green switching policies at intersections of 2 one-way streets. Simulation has shown that implementation of this policy significantly reduces the overall average delay of vehicles at the controlled intersections.

It is important to mention that, the abstracted 1-intersection model introduced here and the results obtained in this paper can be extended in order to obtain coordination strategies for networks with more complicated signalized intersections. Nevertheless, this is beyond the scope of this paper, and it will be dealt with in a forthcoming one.

This paper is organized as follows. Section 2 provides an overview on the currently implemented approaches to control

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 224498.

This work was partially supported by projects CICYT and FEDER DPI2006-15390.

Vázquez and Silva are with Dep. de Informática e Ingeniería de Sistemas, Centro Politécnico Superior, Universidad de Zaragoza, E-50018 Zaragoza, Spain {cvazquez,silva}@unizar.es. Sutarto and Boel are with SYSTeMS, Universiteit Gent B-9052 Zwijnaarde, Belgium, {Herman.Sutarto,Rene.Boel}@UGent.be

of traffic lights, and explains how the model developed in this paper can help in improving the performance of a traffic signal coordinator. Section 3 introduces continuous Petri nets and a simple intersection model. Section 4 introduces an improved 1-intersection model, while section 5 uses this to develop an optimal control law. Section 6 shows via simulations how the proposed control law leads to improvements. Section 7 provides some conclusions, and suggests some future work.

II. Relation to Classical Traffic Engineering Models

Switching strategies for traffic lights [8] can be classified according to how fast they respond to on-line traffic measurements. *Pre-timed* controllers select a fixed period, called the cycle time, for all traffic lights in a given area, and select the fraction of green for each direction in each intersection. Moreover the relative time offsets - the phase shifts between the switching times at consecutive intersections - is selected so as to guarantee as much as possible a green wave of vehicles that do not have to stop. These pre-timed controllers may be adjusted during the day to adapt to different traffic patterns, but they only use the average traffic flow data, not the on-line measurements of queue sizes or of current traffic flows.

Actuated traffic signals react to local measurements of queues of waiting vehicles (or detected oncoming traffic) by switching to green, if at all possible. This may reduce the local average delays significantly, but it is clear that this strategy can destroy any advantage of a green wave. Traffic responsive systems therefore must try to adapt to instantaneous information on the local traffic intensity by a coordinated action for the different traffic lights in a given area. Examples of such strategies have been developed since the 1980s (for example the SCATS [9] and the SCOOT systems [10]). These systems adjust from time to time the green fraction at each intersection, and/or the phase shifts, according to some heuristically developed rules.

Recently the availability of on-line measured traffic data, and the enhanced capabilities of road side control agents, have improved so much that it makes sense to try to develop model based on-line optimization algorithms for coordinated traffic responsive control systems. This paper provides a model that can serve this purpose. The model uses hybrid Petri nets in order to represent the queueing behavior of vehicles at intersections. It allows to optimize the green-red periods of the corresponding traffic lights. Furthermore, by adding models that represent the traffic flow along the links connecting intersections, the behavior of a complete urban traffic network can be described. This idea is advanced in the last section of this paper but the details are left for a forthcoming paper. The next section explains one such hybrid Petri net model for a traffic intersection in an urban area.

III. Continuous Petri Nets

We assume that the reader is familiar with *PN*'s (see, for instance, [11]). A (discrete) *Petri* net PN is a tuple $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$, where P is a finite set of places, T is a finite set of transitions with $P \cap T = \emptyset$, $\mathbf{Pre}$ and $\mathbf{Post}$ are $|P| \times |T|$ sized, natural valued, *pre- and post- incidence matrices*. We assume that $\mathcal{N}$ is connected and that every place has a successor, i.e., $|p^\bullet| \geq 1$ (the set of the input (output) nodes of $v \in P \cup T$ is denoted as ${}^\bullet v$ ($v^\bullet$)). A PN system is defined as $\langle \mathcal{N}, \mathbf{M}_0 \rangle$ with $\mathbf{M}_0 \in \mathbb{N}^{|P|}$. The evolution of the PN system is determined by the firing of its transitions. A transition t_j is said *enabled* at $\mathbf{M}$ iff for every $p_i \in^\bullet t_j$, $\mathbf{M}(p_i) \geq \mathbf{Pre}(p_i, t_j)$, and its *enabling degree* is defined as $Enab(t_i, \mathbf{M}) = \min_{p_i \in {}^\bullet t_j} \lfloor \mathbf{M}(p_i) / \mathbf{Pre}(p_i, t_j) \rfloor$. The firing of t in a certain positive integer amount $\alpha \leq Enab(t, \mathbf{M})$ leads to a new marking $\mathbf{M}' = \mathbf{M} + \alpha \cdot \mathbf{C}(t)$, where $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ is the token-flow matrix.

Time delays can be associated to the firing of transitions, obtaining thus a T-*timed Petri net*. In this paper, only two different kind of delays are used: *deterministic* time delays and *exponentially distributed* random time delays. In the first case, transitions fire after a fixed delay during which it remains enabled. In the second case, the delay is no longer fixed, but it is characterized as a random variable (r.v.) with exponential p.d.f., i.e., if a server of an exponentially distributed transition t_i is enabled at time τ_0, then it *must* fire at time $\tau_0 + \theta_i$ (if it remains enabled during $(\tau_0, \tau_0 + \theta_i)$), where θ_i is a r.v. having an exponential p.d.f. with constant parameter λ_i (the average service rate of each server of t_i).

In order to overcome the state explosion problem and to simplify the analysis, PN models can be *relaxed* by means of fluidification ([12], [13]). According to this, given a PN system, it is transformed into a new model by keeping the same structure and initial marking, but defining a new firing rule for some transitions, which are now called *fluid* or *continuous*. At these fluid transitions, the firing is no longer restricted to be integer valued but can occur in nonnegative real amounts, leading to a real valued marking $\mathbf{m} \in \mathbb{R}_{\geq 0}^{|P|}$. In detail, the enabling of a fluid transition is defined as $enab(t_i, \mathbf{m}) = \min_{p_i \in {}^\bullet t_j} \mathbf{m}(p_i) / \mathbf{Pre}(p_i, t_j)$ (without rounding to the nearest lower integer value). A fluid transition t_i can be fired in any *nonnegative real amount* $\alpha \in [0, enab(t_i, \mathbf{m})]$ leading to $\mathbf{m}' = \mathbf{m} + \alpha \cdot \mathbf{C}(t)$.

There are different ways for introducing time in continuous PNs, the two most important being *infinite server semantics* or *variable speed*, and *finite server semantics* or *constant speed*. Here *infinite server semantics* (ISS) will be considered for relaxing discrete transitions with exponential random delays, since for high populated systems (places containing large amount of tokens), the flow of a continuous transition under ISS approximates well the throughput of an exponentially timed discrete transition [14]. Under ISS the flow (the instantaneous firing speed) through a timed continuous transition t_i is the product of the rate λ_i (equivalent to the parameter of the exponential p.d.f. of transition t_i in the original discrete PN) and $enab(t_i, \mathbf{m})$. For the flow to be well defined, every fluid transition must have at least one input place. If all the transitions in a PN model are fluidified, then the resulting model is called *continuous* PN. On the other hand, if only some transitions are fluidified then the model

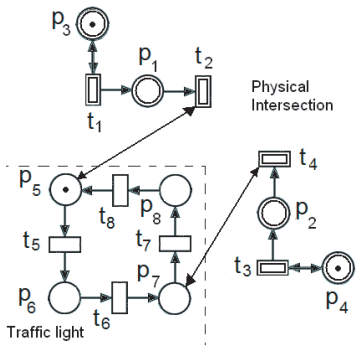

Fig. 1. PN model of an intersection of two one-way streets.

obtained is called a *hybrid* PN. The evolution of a *hybrid* PN can be described, in discrete time with a sampling time $\delta\tau$, by the following difference equation:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \mathbf{C} \cdot \mathbf{\Delta}\sigma_k + \mathbf{C} \cdot \mathbf{\Lambda} \cdot \mathbf{enab}(\mathbf{m}_k) \cdot \delta\tau \quad (1)$$

where $\mathbf{\Delta}\sigma_k$ is the firing count vector of the discrete transitions that fire during the time interval $[\tau_0 + k \cdot \delta\tau, \tau_0 + (k+1) \cdot \delta\tau)$ (its entries related to continuous transitions are null), $\mathbf{enab}(\mathbf{m}_k)$ is a vector whose elements are the enabling degrees of the continuous transitions (its entries related to discrete transitions are null) and $\mathbf{\Lambda}$ is a diagonal matrix whose elements are the corresponding firing rates (i.e., the parameters of the exponential p.d.f.'s in the original discrete PN).

For instance, consider the PN system of fig. 1. This model represents the intersection of two one-way streets controlled by a traffic light, under free-flow condition (i.e., without congestion, the traffic flux is proportional to the traffic density). The subnet $\{p_1, p_2, p_3, p_4\}$ corresponds to the *physical intersection*: tokens in places p_1 and p_2 represent the cars at the queues waiting to enter the intersection while tokens at p_3 and p_4 are the servers at the input transitions (lanes upstream of the queues in p_1 and p_2, respectively). The subnet $\{p_5, p_6, p_7, p_8\}$ represents the *traffic light*. According to this model, the first queue (p_1) is served (t_2 is enabled) only if in the model of the traffic light there is a token at p_5. In a similar way, the second queue (p_2) is served only if in the model of the traffic light there is a token at p_7. A token at either place p_6 or p_8 represents a yellow period (yellow for one queue but red for the other), so, no queue is being served when there is a token in p_6 or p_8. Transitions $\{t_5, t_6, t_7, t_8\}$ modeling the green (green for one queue while it is red for the other) and yellow periods of the traffic light are defined as having deterministic time delays. On the other hand, transitions $\{t_1, t_2, t_3, t_4\}$, corresponding to car arrivals and departures (services) at the intersection, are defined as having exponentially distributed random time delays.

The PN system of fig. 1 is relaxed into a *hybrid* PN model, in which transitions $\{t_1, t_2, t_3, t_4\}$, corresponding to car arrivals and departures (services), are defined as fluid, while other transitions remain discrete (traffic light). Average delays of transitions are defined as $(1, 1/3, 1, 1/3, 20, 5, 20, 5)$. Notice that the *hybrid* model

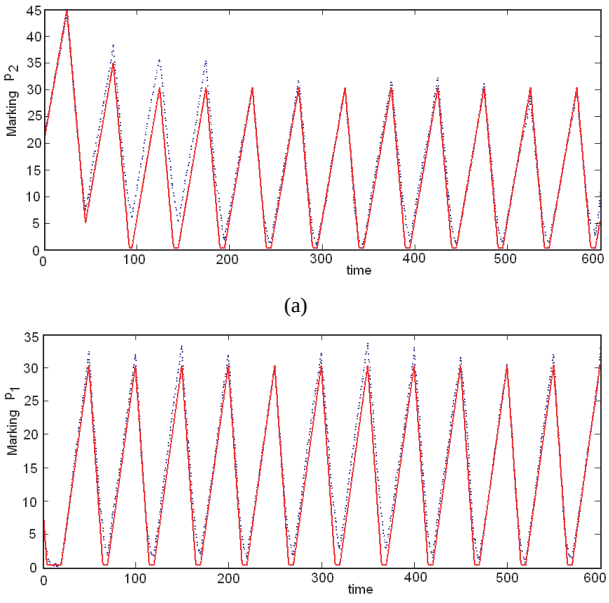

Fig. 2. Marking trajectories of the PN system of fig. 1. Dashed curves correspond to the average trajectory of the stochastic PN, while the continuous ones correspond to the hybrid PN. a) Curves of p_1, b) curves of p_2.

thus obtained is deterministic, since all the stochastic transitions have been fluidified. The marking trajectories of places p_1 and p_2 are shown in fig. 2 (continuous curves). On the other hand, after 20 simulations of the original stochastic discrete model, the average trajectories for the same places were computed, and are also shown in fig. 2 (dashed curves). Notice that the trajectories of both the hybrid PN and the average of the discrete PN models coincide almost perfectly. Hence, the hybrid PN can be used for a quantitative analysis of the original discrete system, with the advantage that this model is deterministic and that the state explosion problem does not appear in this.

IV. The 1-Intersection Model

Lefeber & Rooda [15] studied a mathematical model similar to the hybrid version of fig. 1. They analyze a system consisting of two queues served by a single server. The arrival and services rates are constant but different for each queue. *Setup delays* are considered, i.e., switching the server from serving one queue to serving the other queue takes some fixed time delay, during which no queue is being served. In this way, queues correspond to markings at $\{p_1, p_2\}$ in fig. 1, while the server plays the role of the traffic light.

The results of [15] characterize the optimal steady state periodic orbit that minimizes

$$J = \frac{1}{\mathcal{T}_{ss}} \int_0^{\mathcal{T}_{ss}} [x_1(\tau) + x_2(\tau)] \cdot d\tau \quad (2)$$

where x_1 and x_2 denote the queues (markings at $\{p_1, p_2\}$ in fig. 1) and $\mathcal{T}_{ss}$ denotes the period of the orbit. Furthermore, they also provide a feedback control law that guarantees the convergence of the system to that orbit. That work was

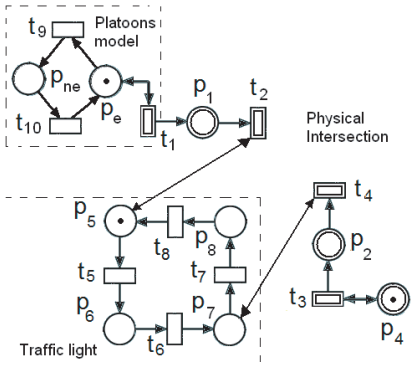

Fig. 3. PN model of an intersection of two one-way streets, the arrivals to the first queue occur as platoons.

the inspiration for considering, in a first step, the model of fig. 1 for optimization in traffic intersections. However, an important assumption is made: the arrival rates are *constant*.

In an urban traffic network, the cars departing one intersection are the arrivals to the neighboring ones. Since each intersection is being controlled by a traffic light, the departures do not have constant flow rate, and so the arrivals to downstream intersections do not occur with a constant flow rate but as *platoons* (*batches* of cars moving closely together). In order to consider those platoon-like arrivals, the PN of fig. 1 is modified, obtaining thus the system of fig. 3.

In this new model, the arrivals to the first queue occur as platoons. A token in place p_e enables t_1, meaning cars arriving with a rate λ_1. After some given delay θ_9, transition t_9 fires (which is discrete and deterministically timed) removing the token at p_e and putting it in p_{ne}, in this way, t_1 is no longer enabled meaning that no cars can arrive. After a given delay θ_{10}, transition t_{10} fires and the token returns to p_e, enabling again t_1. In this way, the time during which a platoon is arriving at queue 1 is θ_9, the number of cars that arrive during that time is $\theta_9 \cdot \lambda_1$, the time between the arrival of two consecutive platoons (between the car leading a platoon and the car leading the next platoon) is $\theta_9 + \theta_{10}$. Therefore, transitions t_9, t_{10} and places p_e and p_{ne} characterize the platoons that arrive to the first queue. In this model, the arrivals to the second queue still occur with a constant rate. Those arrivals can also be generalized in order to consider platoons. However, for the sake of simplicity, they will be maintained as constant in this paper.

The results obtained in [15] do not provide the optimal behavior for the system of fig. 3, since the arrivals to the first queue do not occur with a constant rate. Furthermore, in the case studied in this paper, the discrete subnet (the subnet described by transitions $\{t_5, t_6, t_7, t_8, t_9, t_{10}\}$) does not describe a sequential process but a concurrent one, i.e., there are many possible trajectories in the untimed subnet. For instance, in the PN model of fig. 1, the discrete subnet always evolves with the sequence $t_5, t_6, t_7, t_8, t_5, ...$; but in the model of fig. 3 the discrete subnet can evolve as $t_5, t_9, t_6, t_7, t_8, t_{10}, ..$ or $t_5, t_9, t_6, t_{10}, t_7, t_8, ..$ etc.; the trajectory or sequence that occurs in the timed model depends on the delays of t_5 and t_7, i.e., the parameters to optimize. For this reason, the optimal periodic orbit for this new model is difficult to characterize, because it is not possible to obtain a description of the cost function J (2) in parametric form.

V. Optimization for 1-Intersection

In this section, a parameter optimization problem is introduced and solved for the 1-intersection model of fig. 3, obtaining thus the optimal green periods for the traffic light. Notice that, since the yellow periods are fixed a priori for safety reasons, defining the green periods for each queue the red ones are completely determined, obtaining thus the complete timing for the traffic light.

Given minimum and maximum possible integer values for the time delays of t_5 and t_7 (the green periods), denoted as θ_5^{min}, θ_7^{min}, θ_5^{max} and θ_7^{max} respectively, a finite set of possible control values is defined:

$$CS = \{(\theta_5, \theta_7) \in \mathbb{N} \times \mathbb{N} | \theta_5^{min} \leq \theta_5 \leq \theta_5^{max}, \theta_7^{min} \leq \theta_7 \leq \theta_7^{max}\} \quad (3)$$

Next, given the initial queue lengths ($\mathbf{m}_0(p_1)$ and $\mathbf{m}_0(p_2)$) and a fixed time horizon $\mathcal{T}$, for each pair $(\theta_5, \theta_7) \in CS$, the following cost function is computed:

$$J(\mathcal{T}, \theta_5, \theta_7) = \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \mathbf{w} \cdot \begin{bmatrix} \mathbf{m}(p_1) \\ \mathbf{m}(p_2) \end{bmatrix} \cdot d\tau \quad (4)$$

where $\mathbf{w}$ is a positive row vector representing some optimization weights. Finally, the cost values thus obtained are compared, and so, the minimum of them determines the optimal control policy $(\theta_5^{opt}, \theta_7^{opt})$ to be applied.

Notice that $(\theta_5^{opt}, \theta_7^{opt})$ may not be the optimal values at the steady state. Nevertheless, given the current estimates of the state ($\mathbf{m}_0(p_1)$ and $\mathbf{m}_0(p_2)$), $(\theta_5^{opt}, \theta_7^{opt})$ minimizes the average vehicle delay over an interval of time starting at the present time and looking $\mathcal{T}$ time units into the future (like in *model predictive controllers*).

The main drawback of the previous approach is the high computational cost. However, the computation of J (4) can be achieved very efficiently by computing in parametric form (but off-line) the incremental cost $J(\tau + \Delta\tau) - J(\tau)$, during a time interval $\Delta\tau$ that the system remains in the same discrete state, for each possible discrete state.

For instance, consider the system as in fig. 3. Given firing rates for $\{t_1, t_2, t_3, t_4\}$ as $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$, time delays for $\{t_5, t_6, t_7, t_8\}$ as $(\theta_5, \theta_6, \theta_7, \theta_8) = (20, 5, 40, 5)$ and time delays for $\{t_9, t_{10}\}$ equal to $(\theta_9, \theta_{10}) = (10, 30)$, the system will remain at the same discrete state during $\Delta\tau = min(\theta_5, \theta_9) = 10$ time units (i.e., the minimum time delay of those discrete transitions that are enabled at the current discrete state). During such time $\Delta\tau$, the queues will change, i.e., the marking at p_1 and p_2, but this evolution is deterministic and can be computed in parametric form.

In particular, it is easy to prove that, given initial values $\mathbf{m}_0(p_1)$ and $\mathbf{m}_0(p_2)$, after $\Delta\tau$ time units during which the

discrete state remains at $\mathbf{m}(p_e) = 1$ and $\mathbf{m}(p_5) = 1$, the variables are:

$$\mathbf{m}_{\Delta\tau}(p_1) = \begin{cases} (\lambda_1 - \lambda_2) \cdot \Delta\tau + \mathbf{m}_0(p_1) & \text{if } \Delta\tau \leq \tau_c \\ \frac{\lambda_1}{\lambda_2} + \left[1 - \frac{\lambda_1}{\lambda_2}\right] e^{-\lambda_2(\tau - \tau_c)} & \text{if } \Delta\tau > \tau_c \end{cases}$$
$$\mathbf{m}_{\Delta\tau}(p_2) = \lambda_3 \cdot \Delta\tau + \mathbf{m}_0(p_2) \tag{5}$$

where $\tau_c = (1 - \mathbf{m}_0(p_1))/(\lambda_1 - \lambda_2)$ is the time needed for the first queue to reach the value of 1 (according to the ISS, if the first queue $\mathbf{m}(p_1)$ is larger than 1 then transition t_2 is constrained by p_5, so this queue decreases according to a constant speed, but if the queue is lower than 1 then t_2 is constrained by p_1 and so, in this case, it decreases with a speed that depends on the queue's current value). Furthermore, the increment of the cost function defined as

$$\Delta J = \int_0^{\Delta\tau} \mathbf{w} \cdot \begin{bmatrix} \mathbf{m}(p_1) \\ \mathbf{m}_(p_2) \end{bmatrix} \cdot d\tau = w_1 \int_0^{\Delta\tau} \mathbf{m}(p_1) d\tau + w_2 \int_0^{\Delta\tau} \mathbf{m}(p_2) d\tau \tag{6}$$

can be easily computed by using, for this discrete state, the following expressions:

$$\int_0^{\Delta\tau} \mathbf{m}(p_1) d\tau = \frac{(\lambda_1 - \lambda_2)}{2} \Delta\tau^2 + \mathbf{m}_0(p_1)\Delta\tau \tag{7}$$

if $\Delta\tau \leq \tau_c$. For the case in which $\Delta\tau > \tau_c$ then

$$\int_0^{\Delta\tau} \mathbf{m}(p_1) d\tau = \frac{(\lambda_1 - \lambda_2)}{2} \Delta\tau^2 + \mathbf{m}_0(p_1)\Delta\tau + \frac{\lambda_1}{\lambda_2}(\Delta\tau - \tau_c) + \left[\frac{\lambda_1}{\lambda_2^2} - \frac{1}{\lambda_2}\right]\left[e^{\lambda_2(\tau_c - \Delta\tau)} - 1\right] \tag{8}$$

and in both cases

$$\int_0^{\Delta\tau} \mathbf{m}(p_2) d\tau = \frac{\lambda_3}{2} \Delta\tau^2 + \mathbf{m}_0(p_2)\Delta\tau \tag{9}$$

By following a similar reasoning, expressions for $\mathbf{m}_{\Delta\tau}(p_1)$, $\mathbf{m}_{\Delta\tau}(p_2)$ and ΔJ can be obtained for all the different discrete states. Therefore, the computation of the cost function for a given pair (θ_5, θ_7) can be quickly achieved by following a discrete-event simulation algorithm:

Initialize	$\tau = 0, J_{ac} = 0$
While	$\tau \leq \mathcal{T}$ (time horizon) do
	Compute the remaining time at the current discrete state: $\Delta\tau$.
	Compute the queues at $\tau + \Delta\tau$, i.e., $\mathbf{m}_{(\tau+\Delta\tau)}(p_1)$ and $\mathbf{m}_{(\tau+\Delta\tau)}(p_2)$ by using the expressions obtained off-line (5).
	Compute the incremental cost: ΔJ, by using the expressions obtained off-line (6).
	Add the incremental cost: $J_{ac} = J_{ac} + \Delta J$.
	Update the time: $\tau = \tau + \Delta\tau$.
	Fire the enabled discrete transition.
end	
The total	cost function is given by: $J(\tau) = \frac{1}{\tau} \cdot J_{ac}$

In this way, the optimization can be achieved efficiently by enumerating the small number of discrete states. For instance, fig. 4 shows the results obtained for the system of

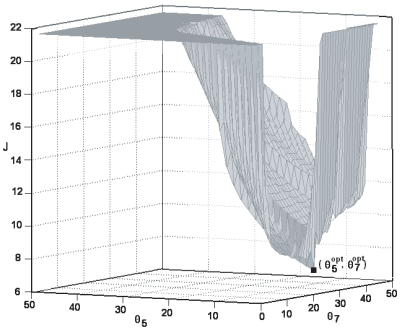

Fig. 4. Computation of optimal green periods: $(\theta_5^{opt}, \theta_7^{opt}) = (4, 27)$ with $J = 7.18$.

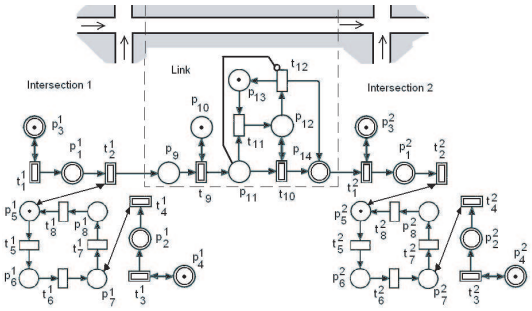

Fig. 5. PN model of 2 intersections connected by a link.

fig. 3 with rates $[1, 3, 1, 3]$ for $\{t_1, t_2, t_3, t_4\}$, delays $[10, 30]$ for $\{t_9, t_{10}\}$ and $[5, 5]$ for $\{t_6, t_8\}$, weights $\mathbf{w} = [1, 1]$ and a time horizon $\mathcal{T} = 1200$. Notice that the minimum value for this, i.e., the optimal green period, is well defined (the dashed square: $(\theta_5^{opt}, \theta_7^{opt}) = (4, 27)$ with $J = 7.18$). Furthermore, the computation time spent for one cost evaluation is fast enough to allow use it in an on-line MPC-like control law (a CPU with a Intel Core 2 Duo at 2GHz spends 84 seconds for computing the optimal control, which is considerably lower than the time horizon $\mathcal{T} = 1200$).

VI. Controlling 1-intersection in a traffic network

This section is devoted to advance some ideas leading to a control strategy for urban traffic networks by using the optimization algorithm introduced in the previous section.

The proposed example is shown in fig. 5. This system consists of 2 intersections connected by a link (one-way street). The first intersection is modeled as in fig. 1, i.e., the incoming flows occur with constant rates. The output flow of one direction is connected to a second intersection by a link, which introduces a pure delay. The subnet $\{p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, t_9, t_{10}, t_{11}, t_{12}\}$ defines such link, in which transitions $\{t_{11}, t_{12}\}$ are discrete and $\{t_9, t_{10}\}$ are continuous. This link model works as follows: when a platoon is departing from intersection 1, whose output transition is t_2^1, tokens flow to p_9 and then through t_9 into place p_{11}, where they are accumulated, enabling at the same time t_{11}. After θ_{11} time units, t_{11} is fired (θ_{11} is the time needed by the leading vehicle of the platoon leaving p_1^1

before reaching the downstream intersection), marking p_{12}, and enabling t_{10}, so that the platoon is free to follow its way towards the intersection 2 (i.e., from p_{11} to place p_{14} and then to p_1^2, which is the queue at the second intersection). When the last token is gone from p_{11} (the last car has left the link), t_{12} is enabled and then fired, resetting the initial condition of this part of the model.

The dynamic behavior of the second intersection can be represent by means of the model of fig. 3, since the incoming flow through t_1^2 occurs as platoons, while the incoming flow through t_3^2 occurs with a constant rate. In this way, transitions $\{t_1^2, t_2^2, t_3^2, t_4^2\}$ and places $\{p_1^2, p_2^2, p_4^2\}$ of fig. 5 correspond to $\{t_1, t_2, t_3, t_4\}$ and $\{p_1, p_2, p_4\}$ of fig. 3, respectively (in the same order). In a similar way, the nodes modeling the traffic light of the second intersection in fig. 5 $\{p_5^2, t_5^2, p_6^2, t_6^2, p_7^2, t_7^2, p_8^2, t_8^2\}$ correspond to nodes $\{p_5, t_5, p_6, t_6, p_7, t_7, p_8, t_8\}$ in fig. 3. The information about arriving platoons (i.e., delays of $\{t_9, t_{10}\}$ and marking of $\{p_e, p_{ne}\}$ in fig. 3) is not directly available from the 2-intersections model, thus it must be obtained by off-line computation or on-line estimation.

It is assumed that the periods of the traffic light for the first intersection are fixed. The goal in this example is to optimize the periods of the second traffic light, showing that the model for 1-intersection of fig. 3 can capture the interactions with neighbor intersections in a urban network. An MPC controller is implemented for this, by using the optimization algorithm introduced in the previous section. Let us describe this:

1. The parameters of the arriving platoons from intersection 1 are estimated. These parameters are incorporated into the 1-intersection model of fig. 3. The other rates and markings are given by the corresponding ones of intersection 2.
2. The optimization algorithm introduced in the previous section is used for computing optimal green periods for the local traffic light, using some fixed finite horizon $\mathcal{T}$.
3. Those green periods are applied to the system.
4. After a fixed time $\mathcal{T}_{upd}$ (updating time), lower than the horizon $\mathcal{T}$, return to the step 1, i.e., estimate again the parameters of incoming platoons and compute and apply again the corresponding optimal green periods. While the time horizon $\mathcal{T}$ can be large enough to consider a few traffic-light cycles, the updating time $\mathcal{T}_{upd}$ must be small enough in order to update the estimation of the incoming platoons, and thus, to compute again the optimal periods with the most accurate available information.

Since the rates of the transitions of intersection 1 are assumed to be fixed and known, the parameters of the platoons arriving to intersection 2 are constant, and can be computed off-line. Nevertheless, in a general urban traffic network where the periods of several traffic lights are being adjusted, those parameters are variable and it is required to estimate them in real time. For such case, step 4 in the previous control procedure states that the estimation, optimization and modification of the traffic light periods has to be iterated periodically.

This control strategy was applied to the hybrid model of fig. 5. Delays for the first traffic light $\{t_5^1, t_6^1, t_7^1, t_8^1\}$ are $(20, 5, 20, 4)$ (in the same order). Delays for the intersection 1 $\{t_1^1, t_2^1, t_3^1, t_4^1\}$ are $(1, 1/3, 1/3, 1/5)$, for the second intersection $\{t_1^2, t_2^2, t_3^2, t_4^2\}$ are $(1/3, 1/5, 1, 1/3)$, while delays for the link $\{t_9, t_{10}, t_{11}, t_{12}\}$ are $(1/10, 1/3, 30, 1/3)$ (the link delay is $\theta_{11} = 30$). The initial queues are given by $(10, 20)$ for $\{p_1^1, p_2^1\}$ and $\{p_1^2, p_2^2\}$, respectively. During the experiment, the control law was applied to intersection 2, while the periods for the traffic light of intersection 1 are fixed. The optimal control law was computed and applied each 5 time units (the computation of the control law takes 3.62 seconds on a CPU with Intel Core 2 Duo at 2.00GHz each time it is computed), with an horizon of 110 time units. The parameters of arriving platoons are obtained from an estimation procedure (not described in this work, but these can be easily computed off-line for this example). The results are shown in fig. 6. The marking at p_1^2 corresponds to the queue whose incoming flow occurs in platoons. The square signals in the lower part of fig. 6 correspond to the green period (dashed line) of the traffic light for that queue and the incoming platoons (continuous line, cars are added to the queue when this signal is 1). As it can be seen, the controller synchronizes the green period with the incoming platoons in order to induce a green wave, reducing thus the queue at p_1^2 as much as possible. The value obtained for the cost function (defined as in (4) with $\mathbf{w} = [1, 1]$ and $\mathcal{T} = 600$) was 11.59, which is considerably lower than the value for the system with fixed green periods 16.59 (without control).

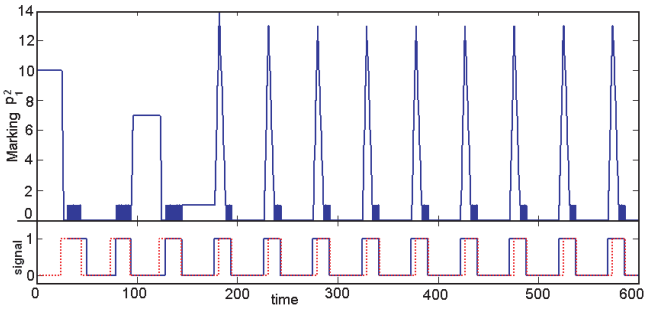

Fig. 6. Marking of queues at the second intersection (fig. 5) under control, and signals of the green period (dashed line) and incoming platoons (continuous line).

VII. Conclusion and Future Work

This paper introduces a PN model for intersections in a urban traffic network. It was shown that this model leads to significant improvement in the traffic performance at a signalized intersection, by using control laws for the corresponding traffic light signals. In the future, the ideas introduced in this paper will be extended in order to simultaneously control, in a *distributed* way, several traffic lights of intersections connected in a urban traffic network.

References

[1] Lämmer S. & Helbing D. (2008). Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical mechanics: Theory and Experiment*, P04019.

[2] Kumar P.R. & Seidman T. I. (1990). Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, vol. 35, 289–298.
[3] Boillot F., Blosseville J.M., Lesort J.B., Motyka V., Papageorgiou M. & Sellam S. (1992). Optimal signal control of urban traffic networks. *6th International Conference on Road traffic Monitoring and Control*, vol. 335, 75–79.
[4] Porche I., Sampath M., Sengupta R., Chen Y.-L. & Lafortune S. (1996). A decentralized scheme for real-time optimization of traffic signals. *In Proc. 1996 IEEE International Conference on Control Applications*, 582–589.
[5] Camponagara E. & Kraus W. (2003). Distributed learning agents in urban traffic control. *In: Progress in artificial intelligence (6th EPIA)*, 324–335.
[6] van den Berg M., De Schutter B., Hegyi A. & Hellendoorn J. (2004). Model Predictive control of mixed urban and freeway networks. *Proc. of the 83rd annual meeting of Transportation Research Board*, Paper 04-3327.
[7] Dotoli M., Fanti M.P. & Meloni C. (2006). A signal timing plan formulation for urban traffic control. *Control Engineering Practice*, vol. 14, 1297–1311.
[8] USDOT: Federal Highway Administration. *Traffic Control Systems Handbook.*
[9] Lowrie P.R. (1982). SCATS principles, methodologies, algorithm. *IEE Conf. on Road traffic Signal*, IEE Publication 207, 67–70.
[10] Hunt P.B., Robertson D. I., Bretherton R.D. & Royle M.C. (1982). The SCOOT online traffic signal optimization technique. *Traffic Engineering and Control*, vol. 23, 190–192.
[11] Silva M. *Introducing Petri Nets. In Practice of Petri Nets in manufacturing*, Chapman & Hall, 1-62.
[12] Alla H. & David R. (1998). Continuous and hybrid Petri Nets. *Journal of Circuits, Systems and Computers*, vol. 8(1), 159-188.
[13] Silva M. & Recalde L. (2004). On fluidification of Petri net models: from discrete to hybrid and continuous models. *Annual Reviews in Control*, vol. 28(2), 253–266.
[14] Vázquez C.R., Recalde L. & Silva M. (2008). Stochastic–Continuous State Approximation of Markovian Petri Net Systems. *Procceding of the 47th IEEE Conference on Decision and Control.*
[15] Lefeber E. & Rooda J.E. (2006). Controller design for switched linear systems with setups. *Physica A*, vol. 363, 48–61, 2006.