MODELLING AND CONTROLLING TRAFFIC BEHAVIOUR WITH CONTINUOUS PETRI NETS

Jorge Júlvez [*,1] **René Boel** [**]

[*] *Departamento de Informática e Ingeniería de Sistemas*
Universidad de Zaragoza, María de Luna 3, E-50015
Zaragoza, Spain, julvez@unizar.es
[**] *SYSTeMS, Universiteit Gent*
B-9052 Zwijnaarde, Belgium, Rene.Boel@UGent.be

Abstract: Traffic systems are discrete systems that can be heavily populated. One way of overcoming the *state explosion* problem inherent to heavily populated discrete systems is to relax the discrete model. Continuous Petri nets (PN) represent a relaxation of the original discrete Petri nets that leads to a compositional formalism to model traffic behaviour. This paper introduces some new features of continuous Petri nets that are useful to obtain realistic but compact models for traffic systems. Combining these continuous PN models with discrete PN models of traffic lights leads to a hybrid Petri net model that is appropriate for predicting traffic behaviour, and for designing traffic light controllers that minimize the total delay of the vehicles in the system.

Keywords: Petri-nets, Road traffic, Traffic Control, Models, Continuous Systems

1. INTRODUCTION

The aggregate behaviour of vehicles in a traffic network requires a model with various modes of operation such as free flow traffic, saturated/congested traffic, traffic jams, stop-and-go waves, etc. The use of traffic models gives one the chance of analyzing, predicting and controlling the behaviour of traffic systems. The application of control strategies allows one to improve some interesting traffic performance measures such as throughput, delay and fuel consumption.

The population of a traffic system is described by the number of vehicles, a discrete value. Hence discrete event models can accurately describe the behaviour of traffic systems. However, it is well known that highly populated discrete systems suffer from the state explosion problem. One way of overcoming this problem is to relax the original model. In the scope of traffic systems, macroscopic models (see (Hoogendoorn and Bovy, 2001; Kotsialos *et al.*, 2002; Helbing, 1997)) disregard individual cars and consider mainly three real valued variables: density, speed and flow.

This paper introduces an aggregated model for traffic systems based on continuous/hybrid Petri nets. The road network to be modelled is subdivided into sections. Each section is modelled separately by a continuous Petri net. The model for the whole network is obtained by joining the nets for the sections: The flow of cars from one section to the next section is represented by the flow of the transition interconnecting both sections. Thus, the model becomes highly compositional. In the proposed model the flow through transitions is variable (similar to (Tolba *et al.*, 2001)), i.e., infinite servers semantics is used, as opposed to earlier papers (Febbraro and Sacone, 1998; Febbraro *et al.*, 2001) using Petri nets with constant speed of transitions, i.e., finite servers semantics. Under infinite servers semantics the flow of a transition is proportional to its enabling degree. This fact entails that the rising edge of the *fundamental traffic diagram* can be modelled more realistically with infinite servers semantics than with finite

[1] Supported by D.G.A. ref B106/2001 and a European Community Marie Curie Fellowship, CTS, contract number: HPMT-CT-2001-00278

servers semantics. Traffic lights are modelled by discrete places and transitions. The control of the system can be achieved by determining the switching times of the traffic lights.
The rest of the paper is organized as follows: Section 2 introduces continuous Petri nets and some modelling considerations. In Section 3, some special features of continuous PNs are introduced. Section 4 makes use of these features to obtain a more realistic representation of traffic systems. A control scenario is considered in Section 5. Conclusions are drawn in Section 6.

2. CONTINUOUS PETRI NET SYSTEMS

2.1 Timed Continuous Petri Net Systems

The reader is assumed to be familiar with Petri nets (PNs) (see for example (Silva, 1993)). The Petri net systems that will be considered here are *continuous* (Júlvez, 2004). Continuous PNs are obtained as a relaxation of *discrete* ones. Unlike "usual" discrete systems, the amount in which a transition can be fired in a continuous Petri net is a nonnegative real number; the state of a continuous PN at instant τ is a vector $\mathbf{m}(\tau)$ of nonnegative real numbers. Graphically, a continuous place is represented as a double circle and a continuous transition as a white box. Matrices **Post** and **Pre** are the arc weight matrices and $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$ is the token flow matrix.
For the timing interpretation, infinite servers semantics (Recalde and Silva, 2001) (or variable speed) will be used. According to this semantics the flow through transition t at instant τ is defined as $\mathbf{f}[t](\tau) = \boldsymbol{\lambda}[t] \cdot \text{enab}(t, \mathbf{m}(\tau))$ where $\boldsymbol{\lambda}[t] > 0$ is the constant internal speed of the transition and $\text{enab}(t, \mathbf{m}(\tau)) = \min_{p \in {}^\bullet t}\{\mathbf{m}[p](\tau)/\mathbf{Pre}[p, t]\}$ is the enabling degree of the transitions. Thus, the flow of a transition is proportional to the marking of the input place determining the enabling degree. The evolution of the marking is given by $\dot{\mathbf{m}}(\tau) = \mathbf{C} \cdot \mathbf{f}(\tau)$.

2.2 Some modelling considerations

Infinite servers semantics allows system models in which the processing speed, i.e., flow of transitions, is proportional to the number of customers in the upstream place, i.e., enabling degree. The following examples show how the flow of transitions and the rate of change of the marking of places can be affected by the arc weights.
Consider transition t_1 (see Figure 1(a)) that has one input place p_1. Its flow is $\mathbf{f}[t_1] = \boldsymbol{\lambda}[t_1] \cdot \mathbf{m}[p_1]/z$ where $z > 0$ is the weight of the arc. As shown above under infinite servers semantics the marking changes according to $\dot{\mathbf{m}}[p_1] = -z \cdot \mathbf{f}[t_1] = -\boldsymbol{\lambda}[t_1] \cdot \mathbf{m}[p_1]$. Thus, the evolution of the marking of p_1 does not depend on z.
By manipulating the system in Figure 1(a) it is possible to obtain a system in which the evolution of p_1 depends on the weight of its input (output) arc. Consider the system in Figure 1(b) with

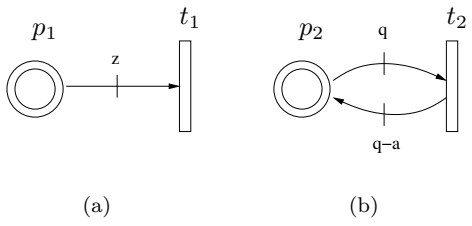


Fig. 1. Two modelling options.

$q > 0$ and $q - a > 0$ since arc weights must be positive. The flow of the transition is $\mathbf{f}[t_2] = \boldsymbol{\lambda}[t_2] \cdot \mathbf{m}[p_2]/q$ and the marking of p_2 evolves according to $\dot{\mathbf{m}}[p_2] = (q - a - q) \cdot \mathbf{f}[t_2] = -a/q\boldsymbol{\lambda}[t_2] \cdot \mathbf{m}[p_2]$, depending on the parameter values q and a. If $a > 0$ the marking of p_2 decreases (due to the constraint $q - a > 0$ the maximum rate of decrease is bounded by $\dot{\mathbf{m}}[p_2] = -\boldsymbol{\lambda}[t_2] \cdot \mathbf{m}[p_2]$). If $a = 0$ then $\mathbf{m}[p_2]$ is constant. If $a < 0$ then $\mathbf{m}[p_2]$ increases.

3. SPECIAL FEATURES OF THE MODEL

This section proposes two modifications to the continuous Petri net formalism presented in Section 2 that are useful to obtain more realistic, and yet compact, models of traffic behaviour. The standard infinite servers semantics represents instantaneous flow of material (or vehicles in the traffic model) from one place to the next one. This is impossible in real systems because vehicles have a finite speed. Moreover, places without input flow empty exponentially slowly over an infinite interval of time according to infinite servers semantics. Again this is not realistic for many applications. This section shows how the model can be modified to remedy these drawbacks.

3.1 Discrete time model

The system in Figure 2 represents a machine, t_1, working at constant speed, $\mathbf{f}[t_1] = \boldsymbol{\lambda}[t_1] \cdot \mathbf{m}[p_1]$, that places its production in a conveyor, p_2. One can imagine that machine t_1 places pieces of finished material at uniformly distributed locations on the conveyor belt p_2, that moves those pieces to the second machine t_2. Machine t_2 processes its input material and stores it in the warehouse p_3. The initial marking of the system is $\mathbf{m_0} = (1\ 0\ 0)$, i.e., the conveyor and the warehouse are initially empty.

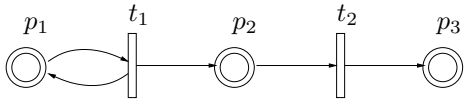


Fig. 2. A continuous Petri net modelling a conveyor.

According to the continuous time model proposed in Section 2 the initial flow of t_1 is $\mathbf{f}[t_1](\tau = 0) = \boldsymbol{\lambda}[t_1]$. This implies that material is placed in the conveyor p_2 from the initial instant $\tau = 0$ ($\mathbf{m}[p_2](\tau) > 0$ for every $\tau > 0$). This entails

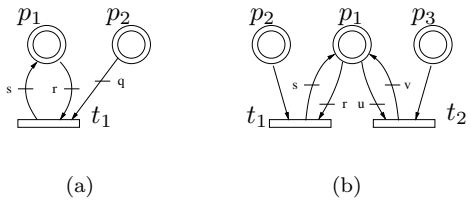
$\mathbf{f}[t_2](\tau) > 0$ for every $\tau > 0$. This behaviour cannot be a faithful representation of a real system behaviour since it implies that the material has spent zero units of time to reach t_2.

In (Corriga *et al.*, 1997) a new net element called *continuous transition with delay arc* is introduced in order to avoid *infinitely fast movements* of material. In this paper, instead of introducing a new element, infinitely fast movements are avoided by considering a discrete time model: Time is discretized in steps (intervals) of length $\Delta > 0$. At the beginning of each step the flow of the transitions is computed with the usual expression for infinite servers semantics: $\mathbf{f}[t](k) = \boldsymbol{\lambda}[t] \cdot \min_{p \in {}^\bullet t}\{\mathbf{m}[p](k)/\mathbf{Pre}[p,t]\}$ for the k^{th} step. The marking at the next step is defined by $\mathbf{m}(k+1) = \mathbf{m}(k) + \mathbf{C} \cdot \mathbf{f}(k) \cdot \Delta$. This way, the flow of a transition during Δ units of time depends only on the marking of its input places at the beginning of the interval. The interval Δ can be seen as the travelling time (delay) of the material between two transitions (Daganzo, 1995). In Figure 2, Δ is the time the conveyor takes to arrive from t_1 to t_2. Notice that the flow of t_2 is zero during the first interval ($\mathbf{f}[t_2](\tau = 0..\Delta) = 0$ if the system is discrete time, $\mathbf{f}[t_2](\tau) > 0$ for every $\tau > 0$ if it is continuous time).

The value of Δ has to be chosen carefully to represent the delay between transitions. A too large Δ can lead to negative markings since the marking may be linearly decreasing during an interval. Fortunately, it is possible to compute an upper bound for Δ in order to ensure the nonnegativeness of the marking. Such upper bound depends only on the structure of the net (not on the marking). To compute this upper bound, each place will be considered separately. It will be assumed that no input flow is coming into the place and it will be computed how fast it can become empty. The interval required to empty the place that takes the shortest time to become empty is the upper bound.

Let us compute how fast the place p_1 of the system in Figure 3(a) can become empty. The marking of p_1 decreases iff $r > s$, hence only this case is considered. Let us first compute how long it takes to empty p_1 if $\mathbf{m}[p_1]/r \leq \mathbf{m}[p_2]/q$, i.e., $\mathbf{m}[p_1]$ defines the enabling degree of t_1. In that case $\mathbf{f}[t_1](k) = \boldsymbol{\lambda}[t_1] \cdot \mathbf{m}[p_1](k)/r$ and $\mathbf{m}[p_1](k+1) = \mathbf{m}[p_1](k) + (s-r) \cdot \boldsymbol{\lambda}[t_1] \cdot \mathbf{m}[p_1](k)/r \cdot \delta$. It follows that $\mathbf{m}[p_1](k+1) = 0$ when $\delta = r/(\boldsymbol{\lambda}[t_1] \cdot (r-s))$. Notice that in the case that $\mathbf{m}[p_1]/r > \mathbf{m}[p_2]/q$ ($\mathbf{m}[p_2]$ defines the enabling degree) the flow through t_1 would be less than in the previous case and therefore it would take longer to empty p_1. For the system in Figure 3(a), the shortest time to empty p_1 is $r/(\boldsymbol{\lambda}[t_1] \cdot (r-s))$. Any Δ smaller than $r/(\boldsymbol{\lambda}[t_1] \cdot (r-s))$ prevents p_1 from becoming negative.

A similar approach can be taken to compute a bound for Δ for a system having places with several output transitions (see Figure 3(b)). To compute the fastest emptying time of p_1 only the



(a) (b)

Fig. 3. The value of Δ is upper bounded.

output transitions that decrease the marking are considered, i.e., t_1 (t_2) is considered iff $r > s$ ($u > v$). The shortest emptying time occurs when p_1 is determining the flow of both output transitions. For a system with several places, to avoid negative markings the value of Δ has to be at most:

$$\min_{p,\, \exists t \in\, p^\bullet \text{ such that } \mathbf{Pre}[p,t] > \mathbf{Post}[p,t]} \left\{ \frac{1}{\sum_{\substack{t \in\, p^\bullet \\ \mathbf{Pre}[p,t] > \mathbf{Post}[p,t]}} \frac{\boldsymbol{\lambda}[t] \cdot (\mathbf{Pre}[p,t] - \mathbf{Post}[p,t])}{\mathbf{Pre}[p,t]}} \right\}$$

3.2 Emptying places in finite time

Let us consider the discrete time evolution of the system in Figure 4. Let Δ be the length of the time interval of the discrete time model (according to the previous Subsection, Δ is upper bounded by $\min\{1/\boldsymbol{\lambda}[t_1], 1/\boldsymbol{\lambda}[t_2]\}$). After the first interval, the marking of p_1 is $\mathbf{m}[p_1](1) = \mathbf{m}[p_1](0) + \mathbf{C} \cdot \mathbf{f}[t_1](0) \cdot \Delta = \mathbf{m}[p_1](0) - \boldsymbol{\lambda}[t_1] \cdot \mathbf{m}[p_1](0) \cdot \Delta = (1 - \boldsymbol{\lambda}[t_1] \cdot \Delta) \cdot \mathbf{m}[p_1](0)$. After the second interval $\mathbf{m}[p_1](2) = (1 - \boldsymbol{\lambda}[t_1] \cdot \Delta) \cdot \mathbf{m}[p_1](1) = (1 - \boldsymbol{\lambda}[t_1] \cdot \Delta)^2 \cdot \mathbf{m}[p_1](0)$ and after the k^{th} interval $\mathbf{m}[p_1](k) = (1 - \boldsymbol{\lambda}[t_1] \cdot \Delta)^k \cdot \mathbf{m}[p_1](0)$. This way, if $\Delta = 1/\boldsymbol{\lambda}[t_1]$, p_1 becomes empty after the first step and remains empty indefinitely. However, if $\Delta < 1/\boldsymbol{\lambda}[t_1]$ the evolution of $\mathbf{m}[p_1]$ follows a geometric progression and never gets completely empty.

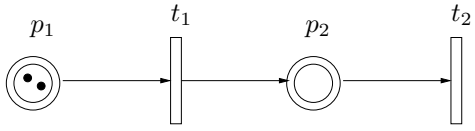


Fig. 4. p_1 is emptied in finite time iff $\Delta = 1/\boldsymbol{\lambda}[t_1]$.

From a modelling point of view a geometrical emptying of a place can be useful, for example, to model how a capacitor discharges exponentially. Nevertheless, for other modelling purposes this feature is not desired. Suppose that the marking of p_1 is the number of customers waiting to be served by t_1, and t_1 is a server that starts working at a speed that is proportional to the length of the queue. If there are no new customers coming into the queue the speed of t_1 should remain constant until the queue empties.

Infinite servers semantics can be modified to avoid falling in a geometric progression when emptying a place: For a given transition t and at a given step k it will be checked whether its input place

determining the enabling degree had input flow (new customers) during the previous step $k-1$. If there was no input flow to that place the flow of t is "forced" to be the same, $\mathbf{f}[t](k) = \mathbf{f}[t](k-1)$, otherwise the usual firing semantics is applied, $\mathbf{f}[t](k) = \boldsymbol{\lambda}[t] \cdot \text{enab}(t, \mathbf{m}, k)$. This way, transitions can be emptied in finite time. This modification in the model leads to *non pure* discrete time infinite servers semantics and could cause negative markings even if the bound for Δ is considered. To avoid negative markings, the flow of the transitions will be forced to be the minimum between the value just described and the flow that would empty one of the input places at the end of the time interval. Thus, places become empty exactly at the end of time intervals.

4. MODELLING TRAFFIC SYSTEMS

This section proposes a model for traffic systems based on the concepts presented in the previous Section. The model requires the road to be divided into several sections. In Subsection 4.1 a continuous PN model of one single section is presented. Subsection 4.2 uses this model as a building block for representing large networks.

4.1 Modelling a road section

The state of a section of a road network is described by three macroscopic variables: Density of cars, average speed and flow. The marking m of a place will represent the number of cars in the section, these cars being uniformly distributed along the length of the section, and having average speed v. Note that m is proportional to the density d of cars along the section. The flow f of cars leaving the section is then $f = d \cdot v$.

In a traffic system the cars in a section with low density travel at a given free speed, *free flow traffic*. Hence, the flow out of the section increases proportionally to the density. When the density of the section is higher, the average speed decreases and the flow out of the section keeps *ideally constant*. If the density is much higher, the traffic becomes heavy and the flow out of the section decreases. This (bell shape) relationship between the flow and the density is known as the *fundamental traffic diagram*. In this Subsection a net that models free flow traffic and constant flow traffic is presented. In the next subsection the decrease in flow due to densities above the traffic jam density is modelled by adding additional flow reducing places.

The number of cars in road section i will be represented by the marking of a place, p_1^i in Figure 5(a), and the flow of cars leaving the section will be the flow of a transition, t_i. If p_3^i is ignored, the use of infinite servers semantics establishes $\mathbf{f}[t_i] = \boldsymbol{\lambda}[t_i] \cdot \mathbf{m}[p_1^i]$, i.e., the outflow is proportional to the density. Hence, the subnet p_1^i, t_i with an appropriate $\boldsymbol{\lambda}[t_i]$ models free flow traffic according to the fundamental traffic diagram. It is interesting to notice that this relationship between the flow and the marking, $\mathbf{f}[t_i] = \boldsymbol{\lambda}[t_i] \cdot \mathbf{m}[p_1^i]$, cannot be modelled with finite servers semantics in which the flow of a transition is independent of the marking of its positively marked input places (Balduzzi *et al.*, 2000; Febbraro *et al.*, 2001).

Constant flow traffic can be modelled by adding p_3^i. The marking of p_3^i is always constant and its physical meaning is that given a $\boldsymbol{\lambda}[t_i]$, the value $\boldsymbol{\lambda}[t_i] \cdot \mathbf{m}[p_3^i]$ is the maximum traffic flow admitted by t_i.

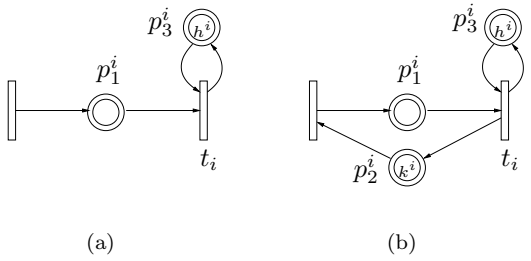


Fig. 5. Modelling a section.

Obviously, the capacity of a road section is finite. This can be modelled by adding a new place to the section model, see p_2^i in Figure 5(b). At any time it holds $\mathbf{m}[p_1^i] + \mathbf{m}[p_2^i] = k^i$ where k^i represents the capacity of the section and $\mathbf{m}[p_2^i]$ the number of free gaps.

4.2 Joining sections

In a Petri net model with several sections, two adjacent sections, i, j, share a transition, t_i, whose flow represents the number of cars passing from section i to section j per time unit. Hence, a given transition t_i of the net model has three input places: p_1^i representing the number of cars in section i, p_3^i with constant marking bounding the flow of t_i and p_2^j representing the number of gaps in section j. Therefore, the flow of cars from section i to section j also depends on the number of gaps in the downstream section j, $\mathbf{f}[t_i] = \boldsymbol{\lambda}[t_i] \cdot \min\{\mathbf{m}[p_1^i], \mathbf{m}[p_3^i], \mathbf{m}[p_2^j]\}$. This fact is very realistic, if one considers for example, how a traffic jam (decreasing of the flow when the density is high) propagates from downstream to upstream sections. The flow of t_i is the minimum between the number of cars desiring to leave the section (sending function) and the number of cars allowed to enter the next section (receiving function) (Daganzo, 1995).

The outflow from a low density section i (free flow) is proportional to the number of cars ($\mathbf{f}[t_i] = \boldsymbol{\lambda}[t_i] \cdot \min\{\mathbf{m}[p_1^i]\}$) being $\boldsymbol{\lambda}[t_i]$ the proportionality constant. If the downstream section becomes full, the outflow is proportional to the number of gaps of the downstream section ($\mathbf{f}[t_i] = \boldsymbol{\lambda}[t_i] \cdot \min\{\mathbf{m}[p_2^j]\}$) with $\boldsymbol{\lambda}[t_i]$ as the proportionality constant. That is, the proportionality constant, $\boldsymbol{\lambda}[t_i]$, is the same under both situations. One way to avoid this fact is to use the arc loops presented in Subsection 2.2. The use of such arc loops allows one to have

different proportionality constants for the density of cars and the number of gaps.
Figure 6 shows a traffic model consisting of three sections and arc loops to control the proportionality constants. With an appropriate $\boldsymbol{\lambda}$, that system can be reduced to an equivalent one with only one arc loop on each transition ($\boldsymbol{\lambda}[t_i]$ is already the proportionality constant either for the density or for the number of gaps).

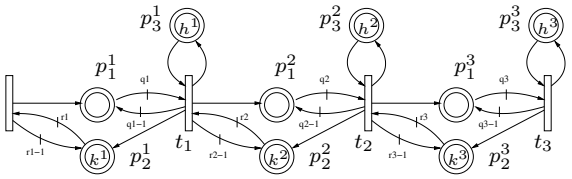


Fig. 6. Traffic system with three sections.

5. CONTROLLING THE SYSTEM

5.1 Modelling traffic lights and intersections

The usual way to control real traffic systems is through traffic lights. A traffic light can be seen as a discrete event system whose state can be either red, amber or green. In our model a traffic light is modelled as a discrete Petri net, see Figure 7 for a traffic light ruling an intersection with two crossing lanes $L1$ and $L2$. Discrete places and transitions are represented by circles and lines.

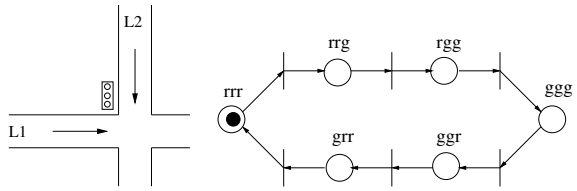


Fig. 7. A discrete Petri net modelling traffic lights in an intersection.

The traffic light has six phases represented by each of the places of the net. A given phase is active when its corresponding place is marked. Since only one phase can be active at a given instant, the number of tokens in the net is 1. The meaning of the phases is: *ggg*: cars of $L1$ crossing, *ggr*: stopping traffic of $L1$, *grr*: cars of $L2$ start crossing, *rrr*: cars of $L2$ crossing, *rrg*: stopping traffic of $L2$, *rgg*: cars of $L1$ start crossing. The use of the phases *ggr*, *grr*, *rrg*, *rgg* allows one a more realistic modelling of the system since they model how the flow of cars softly becomes either zero or positive.
Figure 8 models four sections and an intersection in which the traffic is regulated by a traffic light like the one in Figure 7. It is a hybrid Petri net since it includes discrete and continuous places and transitions. When the traffic light is at *ggg* the flow of t_1 depends on the marking of its input places and the flow of t_2 is 0. Similarly, at phase *rrr* t_1 is blocked and the flow of t_2 depends on the marking of its input places. For the rest of the phases, *ggr*, *grr*, *rrg*, *rgg*, the flow of the transitions involved in the intersection, t_1 and t_2, is defined to model how the flow of cars speeds up and slows down when the traffic light switches: If the traffic light switches to green (*rgg* for section 1) the flow of t_1 increases linearly from zero to the flow computed with the usual infinite firing semantics; if the traffic light switches to red (*ggr* for section 1) the flow of t_1 decreases linearly to zero. These behaviours can be easily simulated by computing a constant flow that produces the marking that would be obtained if the flow increased/decreased linearly.

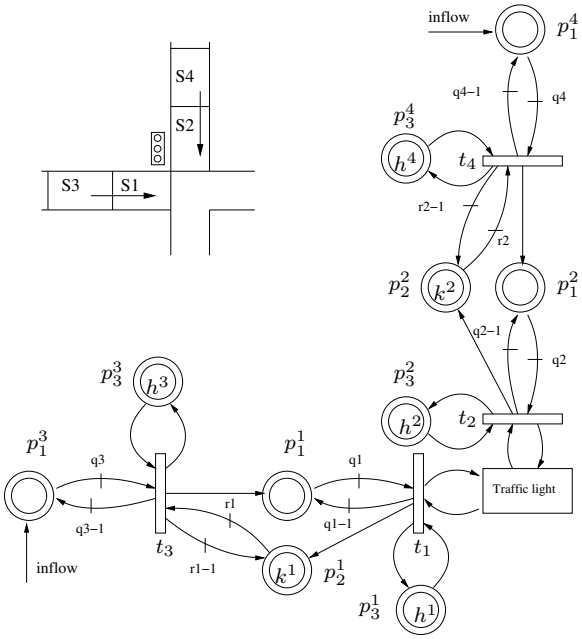


Fig. 8. A Petri net modelling an intersection.

5.2 Objective function

Several objectives can be pursued when controlling a traffic system. In this paper the goal is to minimize the total delay (waiting time) of the cars in the system.
Let us consider the road section in Figure 5(b). For a time step of Δ units of time, the total time spent by the cars in that section is given by $delay = \int_0^{\Delta} \mathbf{m}[p_1^i]d\tau$. Since the flow of the transitions is constant during each time step the evolution of the marking of p_1 is linear and the delay between two steps k and $k+1$ is $\frac{\mathbf{m}[p_1](k)+\mathbf{m}[p_1](k+1)}{2}\cdot\Delta$. Hence, for a given horizon of h steps the delay in that section is $\sum_{k=0}^{h} \frac{\mathbf{m}[p_1](k)+\mathbf{m}[p_1](k+1)}{2}\cdot\Delta$. The total delay of the system is obtained by summing the delays of all the sections.
Recall that the flow of cars crossing the intersection is considered to increase/decrease linearly during phases different to *ggg* and *rrr*. In these cases the expression for the delay is different to the one just presented. Such expression can be computed by solving $delay = \int_0^{\Delta} \mathbf{m}[p_1^i]d\tau$ taking into account the linear evolution of flows.

5.3 Control example

Consider the system in Figure 8. Let us assume that each section is 200 meters long and is composed of two lanes. Let the capacity of the sections be 80 cars ($\mathbf{m}[p_1^i] + \mathbf{m}[p_2^i] = 80$ for $i = 1 \dots 4$), $\boldsymbol{\lambda}[t_i] = 4$ for $i = 1 \dots 4$, $q^i = 100$ for $i = 1 \dots 4$, $r^i = 80$ for $i = 1, 2$, $h^i = 0.5$ for $i = 1 \dots 4$, the initial load of the sections in cars $\mathbf{m}[p_1^1] = 50$, $\mathbf{m}[p_1^2] = 30$, $\mathbf{m}[p_1^3] = 35$, $\mathbf{m}[p_1^4] = 60$ and the time step $\Delta = 8$ seconds. Let us assume that the traffic light is initially red for section 1 and that there exist constant input flows of 0.8 cars per second entering section 3 and 0.5 cars per second entering section 4.

The system will be controlled during 3 traffic light cycles, each of them having a duration of 120 seconds. At the beginning of each cycle the traffic light is forced to become red for section 1. The switching time for each cycle is computed independently. The optimal control for a given cycle is computed by carrying out as many simulations as steps in a cycle: Simulation j switches the traffic light from red to green at the beginning of step $j + 1$. The evolution of the system under optimal control is shown in Figure 9 (in that plot, stars at a level of 10 mean red light for section 1).

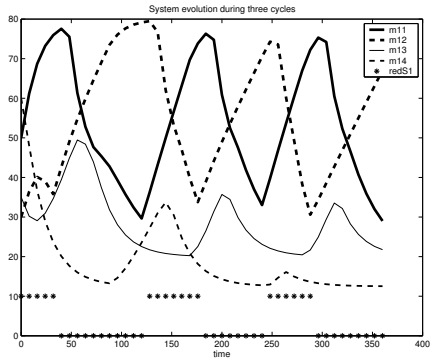


Fig. 9. Evolution of the system in Figure 8 under optimal control.

6. CONCLUSIONS

Several modelling features of continuous Petri nets have been presented. It has been seen that the use of loops of arcs with appropriate weights allows a good and computationally feasible description of the flow of vehicles in a traffic system, allowing control design. A discrete time model is useful to easily model delays in the *material* travelling from one part of the system to another part. A modification in the firing semantics allows places that are not receiving inflow to become empty in a finite time interval. These modelling features are very suitable for traffic systems.

The proposed traffic model is highly compositional: A traffic model can be quickly built by assembling subnets modelling road sections. Since the number of cars in the system is considered a real number, simulation times are independent of the system load.

In order to control the system, discrete Petri nets modelling traffic lights have been introduced. Thus, the net model becomes a hybrid Petri net. A traffic control problem has been presented. Switching times of traffic lights have been computed so that the total delay of the cars in the system is minimized.

REFERENCES

Balduzzi, F., A. Giua and G. Menga (2000). First-order hybrid Petri nets: a model for optimization and control. *IEEE Trans. on Robotics and Automation* **16**(4), 382–399.

Corriga, G., A. Giua and G. Usai (1997). Petri net modeling of irrigation canal networks. In: *Int. Work. on Regulation of Irrigation Canals*. Marrakech, Morocco. pp. 39–48.

Daganzo, C. (1995). A finite difference approximation of the kinematic wave model of traffic flow. *Transportation Research B* **29B**(4), 261–276.

Febbraro, A. Di and S. Sacone (1998). Hybrid Petri nets for the performance analysis of transportation systems. In: *Proc. IEEE Conference on Decision and Control*. Tampa, FL.

Febbraro, A. Di, D Giglio and N. Sacco (2001). Modular representation of urban traffic systems based on hybrid Petri nets. In: *Proceedings of 2001 IEEE Intelligent Transportation Systems*.

Helbing, D. (1997). Traffic data and their implications for consistent traffic flow modelling. In: *Transportation Systems (IFAC, Chania, Greece)* (M. Papageorgiou and A. Pouliezos, Eds.). Vol. 2. pp. 809–814.

Hoogendoorn, S. and P. Bovy (2001). State-of-the-art of vehicular traffic flow modelling. *Special Issue on Road Traffic Modelling and Control of the Journal of Systems and Control Eng. Proc. of the IME I*.

Júlvez, J. (2004). Algebraic Techniques for the Analysis and Control of Continuous Petri Nets. PhD thesis. University of Zaragoza, Spain.

Kotsialos, A., M. Papageorgiou, C. Diakaki, Y. Pavis and F. Middelham (2002). Traffic flow modelling of large-scale motorway using the macroscopic modeling tool metanet. *IEEE Transactions on Intelligent Transportation Systems* **3**(4), 282–292.

Recalde, L. and M. Silva (2001). Petri Nets fluidification revisited: Semantics and steady state. *APII-JESA* **35**(4), 435–449.

Silva, M. (1993). Introducing Petri nets. In: *Practice of Petri Nets in Manufacturing*. pp. 1–62. Chapman & Hall.

Tolba, C., D. Lefebvre, P. Thomas and A. El Moudni (2001). Continuous Petri nets models for the analysis of traffic urban networks. In: *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 2. pp. 1323–1328.