LOADING TRAFFIC ON BOTTLENECKS AN EVENT BASED APPROACH

Nicolas Wagner,
Université Paris Est, LVMT, UMR T9403 INRETS ENPC UMLV.
19 rue Alfred Nobel, Champs sur Marne, 77455 Marne la Vallée Cedex 2, France.
Tel : +33 1 64 15 21 39
Fax: +33 1 64 15 21 40
E-mail: nicolas.wagner@enpc.fr

ABSTRACT

This paper deals with the Dynamic Network Loading Problem. A general formulation is presented for which existence and uniqueness is guaranteed under five physically sound assumptions on the arc travel time models. An efficient computation method is then proposed and applied to networks of bottlenecks. Two important features of the approach are: (1) assuming that demand is described by piecewise linear cumulated volumes on each routes, the computation is exact; (2) time is treated continuously. Numerical illustrations are given.

Keywords: traffic loading, dynamic traffic modelling, bottleneck congestion.

INTRODUCTION

Dynamic traffic assignment aims to find in a network subject to congestion, time-varying traffic volumes on routes that are consistent with the route travel costs. Yet, computing the travel times from given route volumes is both essential, as it's the main step in estimating the travel costs, and challenging, as the problem has both a temporal and network dimension. The problem boils down to derive the traffic volumes on each arc from the route volume vector. It is usually referred to as the *Dynamic Network Loading Problem* (DNLP).

Although Friesz *et al.* (1993) pointed out its importance for the analytical formulation of dynamic assignment models, the literature is quite restricted. Most of the existing solutions rely on simulations. They can be either microscopic (*e.g.* DYNASMART in Mahmassani *et al.* 1995), with an explicit representation of users behaviours on arc and nodes, or macroscopic, the demand being divided into packets of users (Tong and Wong, 2000). Analytical forms of the DNLP are not as frequent. Wu *et al.* (1995) first formulated the loading problem as a system of functional equations and derived a solution method based on finite dimensional approximation. Xu *et al.* (1999) and later Rubio-Ardanaz *et al.* (2003) proposed a radically different approach that can be considered as an event based simulation, and showed it improves significantly the computation speed. Both methods apply to volume-delay travel time models. Yet volume-delay travel times have been shown to be generally unphysical (Daganzo, 1993) and an important number of operational assignment models rather assume bottleneck travel times (*e.g.* Kuwahara and Akamatsu, 1993 or Leurent, 2003). DNLP for the bottleneck model has never been treated explicitly, despite the fact it is regarded as an important category of models combining analytical simplicity, computational robustness, and experimental correctness.

In this paper, we propose a general solution paradigm, inspired by discrete event simulation, and applied it to a network of bottlenecks. The first section is devoted to a formal presentation of the DNLP. The second section presents the global philosophy of the solution method, while the third gives a formal statement of the algorithm. Finally the fourth section gives an example of applications on a network of bottlenecks and numerical experiments.

PROBLEM STATEMENT

Notations

The road network is modelled as an oriented graph (A, N) with N the set of nodes and A the set of arcs a. The set of acyclic routes of (A, N) is denoted R. We aim at formalizing the loading a volume of traffic described as cumulated vehicle flows on each route of the network. In order to do so, let us first define precisely the basic objects of our problem.

Cumulated volumes and instantaneous Flows. Consider a time interval $I = [0; h_M]$, . *Cumulated volumes* of traffic are represented by non decreasing and differentiable almost everywhere function on *I.* They have the following interpretation: $X(h)$ is the number of vehicles that went through a point before h. The set of such functions is denoted $\mathcal{F}(I)$. The following operation is defined on $\mathcal{F}(I)$: *the restriction of a cumulated flow* $X \in \mathcal{F}(I)$ on $[0; h]$ is the function $\tilde{X}$ defined by $\tilde{X}(h') = X(h')$ for $h' \in [0; h]$ and $\tilde{X}(h') = X(h)$. It is denoted $X|_h$.

Instantaneous flows will be systematically denoted by the lower case letter corresponding to their cumulated equivalent.

A vector of cumulated flows $\boldsymbol{Y} = (Y_r)_{r \in R}$ is called a route volume vector and a vector of cumulated volumes $\boldsymbol{X} = (X_a)_{a \in A}$ is called an arc volume vector. The derivative of a cumulated volume at an instant h of I is an instantaneous flow.

Travel time functions and models. Denote $\mathcal{C}(I)$ the set of continuous map from I to $\mathbb{R}_+$. An *arc travel time function* on I is an element of $\mathcal{C}(I)$ and gives the time to go through an arc *a* when entering it at a time $h \in I$. Each arc of the network is endowed with an arc travel time model t_a. An *arc travel time model* t_a is a function taking as input a flow and returning an arc travel time function. In other words, it is a map from $\mathcal{F}(I)$ to $\mathcal{C}(I)$. Physically, an arc travel time model is simply a compact notation for traffic models; given the flow entering into an arc over a period I it allows deducing the resulting travel time over I, denoted $t_a[X]$. The travel time for a departure time h is then denoted $t_a[X](h)$.

Let us assume the following properties on arc travel time models:

Assumption 1. [Continuity]
$t_a: \mathcal{F}(I) \mapsto \mathcal{C}(I)$ is continuous with respect to the uniform norm.

Assumption 2. [No infinite speed]
There exists $t_{min} > 0$ such that for all $X \in \mathcal{F}(I)$ and all $h \in \mathbb{R}$, we have
$t_a[X](h) > t_{min}$.

Assumption 3 [Finiteness]
There exists a continuous map $t_{max}: \mathbb{R}_+ \mapsto \mathbb{R}_+$ such that
$t_a[X](h) \leq t_{max}(X(h_M))$ for all $h \in \mathbb{R}_+$ and $X \in \mathcal{F}(I)$.

Assumption 4 [Strict fifoness]
Let $X \in \mathcal{F}(I)$. The map $h \mapsto h + t_a[X](h)$ is non decreasing. Moreover, for $h_1 < h_2$ in such that $X(h_2) - X(h_1) \neq 0$, we have $h_1 + t_a[X](h_1) < h_2 + t_a[X](h_2)$.

Assumption 5 [Causality]
For all $h \in \mathbb{R}_+$ and $X \in \mathcal{F}(I)$, we have $t_a[X|_h](h) = t_a[X](h)$.

Assumption 1 simply states that a small variation on the cumulated flow on an arc leads to a small variation of the arc travel time. Assumption 2 amounts to say that the time needed to travel along an arc is bounded from below. The finiteness condition (Assumption 3) assumes that if we wait for a sufficient long time, there will be no user left on any arc. The FIFO condition (Assumption 4) states that if two users enter an arc in a given order, they leave it in the same order. Finally, Assumption 5 simply implies that the arc travel time depends on the users that have already entered this arc, but not on the ones that will.

The dynamic network loading problem

Consider a road network (A, N) and endow each arc with a travel time model t_a satisfying Assumptions (1-5). Given a route volume vector $\boldsymbol{Y}$, let us define the resulting volumes on the arcs $\boldsymbol{X}$.

Definition 1. [Outflow of a route volume vector on a network]. The outflow of a route volume vector $\boldsymbol{Y} = (Y_r)_{r \in R}$ is an arc volume vector $\boldsymbol{X} = (X_a)_{a \in A}$ if there exists a collection of route volume vectors $\boldsymbol{Y}_a = \left(Y_{a,r}\right)_{r \in R}$, such that for any a, $X_a = \sum_{r \in R} Y_{a,r}$ that satisfies the following functional system :
$\textit{for every route } r = a_1, \ldots, a_n$

$$\begin{aligned} Y_{a_1,r} &= Y_r \quad (2)\\ Y_{a_i,r} \circ H_{a_{i-1}}\left(X_{a_{i-1}}\right) &= Y_{a_{i-1},r} \quad (3)\\ Y_{a,r} &= 0 \quad (4)\end{aligned}$$

with $H_a[X_a] := id_I + t[X_a]$

The interpretation of Definition 1 is the following. $Y_{a,r}$ is the cumulated flow of vehicles entering a while following route r. In other words the quantity $X_{a,r}(h)$ represents the volume of vehicles following route r that entered arc a before h. Equations (1-3) are essentially volume conservation equations. Equality (1) states that the volume of vehicles entering the first arc of a route r is the number of vehicles entering on route r. Equality (2) expresses that the volume of vehicles entering on arc a_i before h is the volume of vehicles leaving arc a_{i-1} before h. Equality (3) simply states that if a vehicle does not follow a route including arc *a*, it will never go through *a.* As there might be confusion between the cumulated volume entering an arc and following a route $Y_{a,r}$ and the volume on a route Y_r, we will sometimes refer to the latter as the *volumes at origin*.

It has been shown in Meunier & Wagner (2010) that for each route volume vector there is a unique outflow. The DNLP as we understand it in this paper is to find this outflow for a given network and a given route volume vector.

PHILOSOPHY OF THE SOLUTION METHOD

Restrictive assumptions

The proof in Meunier & Wagner is constructive and thus the paper gives an algorithm to load traffic on a road network satisfying to the Assumption (1-5). In a few words the general idea is to proceed recursively. Assume you know a collection of cumulated volumes $(Y_{a,r})_{\mathrm{a} \in \mathrm{A}, \mathrm{r} \in \mathrm{R}}$ satisfying equations (1-3) until the instant h, then by Assumption 2, 4 and 5, you can deduce a new collection cumulated flows $(Y_{a,r})_{\mathrm{a} \in \mathrm{A}, \mathrm{r} \in \mathrm{R}}$ satisfying the same equations until $h + t_{min}$. Assumption 3 guarantees the termination of the recursion. In the general case this algorithm is possibly the most efficient way of solving the problem and in fact in the literature most of the existing algorithms follow more or less this same pattern. Yet by slightly restricting the problem, it can be importantly simplified thus leading to a more efficient solution method.

In this paper only cumulated volumes at origins which are continuous piecewise linear functions of time are considered. In addition, arc travel time models are assumed to lead to piecewise linear travel time functions when applied to continuous piecewise linear route volume vectors.

The primitives manipulated under these assumptions are piecewise linear (PWL) functions. Note that they can easily be encoded under the form of an ordered list of elements $X = (h_i, X_i, x_i)_i$ where X_i is the image of h_i by the PWL function X in and x_i is its derivative on the right. Each element of the list is called a piece. Note that under this formalism it is easy to define the operations of linear combination, composition and inverse. With an adequate implementation there are essentially equivalent to a list traverse and thus in $O(n)$ where n is the number of pieces of the function considered.

Consequences for the loading problem

With the PWL assumptions, the main quantities of the problem are piecewise linear functions of the time. Let us focus on the cumulated volumes $\left(Y_{a,r}\right)_{a\in A, r\in R}$ and consider the set of instants h_i such that there exists a triplet (h_i, y_i, s_i) that belongs to a cumulated volume $Y_{a,r}$. They will be referred to as the *critical instants.* Now assume the cumulated volumes $\left(Y_{a,r}\right)$ are known until h, i.e. that $(Y_{a,r}|_{h)}$ are known. In terms of PWL format it means that all elements (h_i, X_i, x_i) of $Y_{a,r}$ such that $h_i \leq h$ are known. Then if one could find the first critical instant h' after h, as well as the concerned cumulated volume and its new slope, $(Y_{a,r}|_{h'})$ can be deduced and the process can be iterated until completion.

Informally that is just saying that instead of seeing the cumulated functions as functions of the times, we see them as a sequence of transitions from one slope to another occurring at critical instants. The algorithm we proposed is simply to go from critical instants to critical instants and correctly update the values of $(Y_{a,r}|_{h'})$. Our proposition is to formalize this general idea under a discrete event system. Each event corresponds to a change of slope and thus occurs at a critical time.

Example on a simple case

The punctual bottleneck model. We are going to consider a simple example using the punctual bottleneck travel times (see for instance a presentation in Leurent 2003). Let us first introduce its basic equations. Consider the derivation of travel time function $t_a[X]$ from an incoming flow X. Travel along the arc is assumed uncongested except perhaps at a single bottleneck of deterministic capacity k_a located at the exit of the arc. If the entry flow coming in bottleneck has rate in excess of k_a, then a waiting queue develops where users wait to leave queue according to a First In – First Out discipline. Let us define the travel time function by the following relationship, in which $Q(h)$ denotes the number of queued users at h in the bottleneck, and $t_{a,0}$ is the free flow travel time:

$$t_a[X](h) = t_{a,0} + Q(h)/k_a \tag{4}$$

where Q stems from the following differential equation :

$$\frac{dQ}{dh} = \begin{cases} x(h) - k_a & \text{if } Q(h) \neq 0 \text{ or } x(h) - k_a > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

When X is continuous, the resulting travel time $t_a[X]$ is well defined, continuous and differentiable nearly everywhere on I. Also note that when X is a PWL function so is $t_a[X]$.

Resolution on a simple network for piecewise cumulated flows. Assume a simple network of bottlenecks with two origins, O_1 and O_2, two destinations D_1 and D_2 and five arcs denoted a_i, $i = 1..5$. Only two routes are available, $r_1 = a_1, a_3, a_4$ connects O_1 to D_1 while

$r_2 = a_2, a_3, a_5$ connects O_2 to D_2. The only bottleneck is on a_3 with a capacity of $k = 1000$ uvp\hour. The free flow travel times are of arc a_1, a_2 and a_3 are given respectively by $t_0^{a_1} = 1$ hour, $t_0^{a_2} = 1.5$ hour and $t_0^{a_2} = 0$. The network and its characteristic values are depicted in Figure 1.

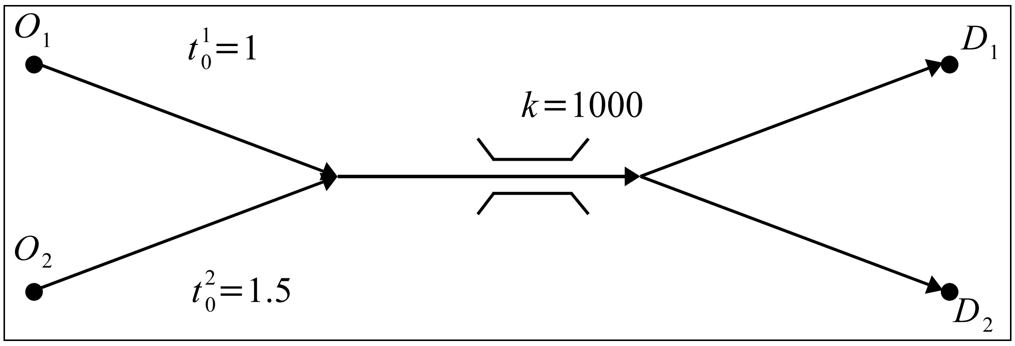

Figure 1 – A simple network

Consider a route volume vector $\boldsymbol{Y} = (Y_{r_1}, Y_{r_2})$ defined on I=$[6{:}00,14{:}00]$. $\boldsymbol{Y}$ is given by its derivative. During $[6{:}00,9{:}00)$, the flow entering r_1 is $y_{r_1} = 1500$ uvp/hour while the flow on route r_2 is $y_{r_2} = 500$ uvp/hour. On $(9{:}00,14{:}00]$ we have $y_{r_1} = y_{r_2} = 250$ uvp/hour.
The resolution can be made iteratively.

1. At 6:00 all the flows of the network are zero except on arc a_1 and a_2 where $x_{a_1} = y_{r_1} = 1500$ uvp/hour and $x_{a_2} = y_{r_2} = 500$ uvp/hour. This stay so until the route flows reaches the end of arc a_1 and a_2. This happens at $h_1 = 6{:}00 + t_0^1 = 7{:}00$ for arc a_1 and at $h_2 = h_A + t_0^2 = 7{:}30$ for a_2.
2. At h_1, the incoming flow on a_3 changes from $x_{a_3} = 0$ to $x_{a_3} = 1500$ uvp/hour. As the flow outgoing from a_3 is bounded by the bottleneck in capacity, there is an exit flow of $x_{a_3}^- = k = 1000$ uvp/hour. A queue began to grow at the rate of $\frac{dQ_{a_3}}{dh} = 1500 - 1000 = 500$ uvp/hour. As all users on a_3 are following route r_1 for the moment, the exit flow is entirely disgorged on arc a_4 and $x_{a_4} = 1000$ uvp/hour.
3. At h_2, the flow entering a_3 switches to $x_{a_3} = 2000$ uvp/hour. The queue growing rate is now $\frac{dQ_{a_3}}{dh} = 2000 - 1000 = 1000$ uvp/hour and the travel time on that arc is $t_{a_3}[X_{a_3}](h_2) = \frac{Q_{a_3}(h_2)}{k} = 0.25$ hour. Consequently the first user following route r_2 to exit arc a_3 will arrive on a_5 at $h_3 = 7{:}45$.
4. At h_3, the new incoming flows of arc a_4 and a_5 are $x_{a_4} = 750$ uvp/hour and $x_{a_5} = 250$ uvp/hour respectively.
5. The change in route flow at $9{:}00$ provokes changes in the incoming flow of arc a_3 at instants $h_4 = 9{:}00$ and $h_5 = 9{:}30$.
6. At h_4, $x_{a_3} = 1250$ uvp/hour and $\frac{dQ_{a_3}}{dh} = 250$. The travel time is now $t_{a_3}[X_{a_3}](h_4) = 1{,}5$. The next change in the exit flow will be at $h_6 = 10{:}30$.
7. At h_5, $x_{a_3} = 500$ and $\frac{dQ_{a_3}}{dh} = -500$. The travel time is now $t_{a_3}[X_{a_3}](h_5) = 1{,}625$. The exit flow will change at $h_7 = 11{:}07$. The queue is now decreasing and will be empty by $h_8 = 12{:}45$.

8. At h_6 the entrance flows on a_4 and a_5 are $x_{a_4} = 1000/3$ and $x_{a_4} = 2000/3$ respectively.
9. At h_7 the entrance flows on a_4 and a_5 are $x_{a_4} = x_{a_5} = 500$ uvp/hour.
10. At h_8, the queue is completely cleared and the entry flows on arc a_4 and arc a_5 corresponds to the exit flow of arc a_1 and a_2 *i.e.* $x_{a_4} = x_{a_5} = 250$ uvp/hour*.*

Figure 2 depicts the solutions of the loading problem by representing the cumulated volume X_{a_3} at the entrance of arc a_3, the cumulated volume Y_{r_1,a_3} at the entrance of arc a_3 following route r_1, the cumulated volume $X_{a_3}^{-}$ at the exit of a_3 (Figure 2, top) and the cumulated flow X_{a_4} at the entrance of arc a_4 (Figure 2, bottom). This example, albeit simple, is quite instructive. First, relatively simple inputs in terms of both networks and route flows can lead to much more complicated arc flows through the interaction at bottlenecks. Second it is reasonably easy to deduce the impacts of a change in an arc incoming flow (*i.e.* an event with our terminology) on the change of flows on the downstream arcs. Informally the mechanism is the following: from each event can be deduced to a sequence of other forthcoming events that needs to be treated chronologically. The main difficulty is to correctly coordinate the actualization of the arc entering flows *i.e.* to handle each event in the right (chronological) order. The algorithm presented in the following essentially addresses this issue.

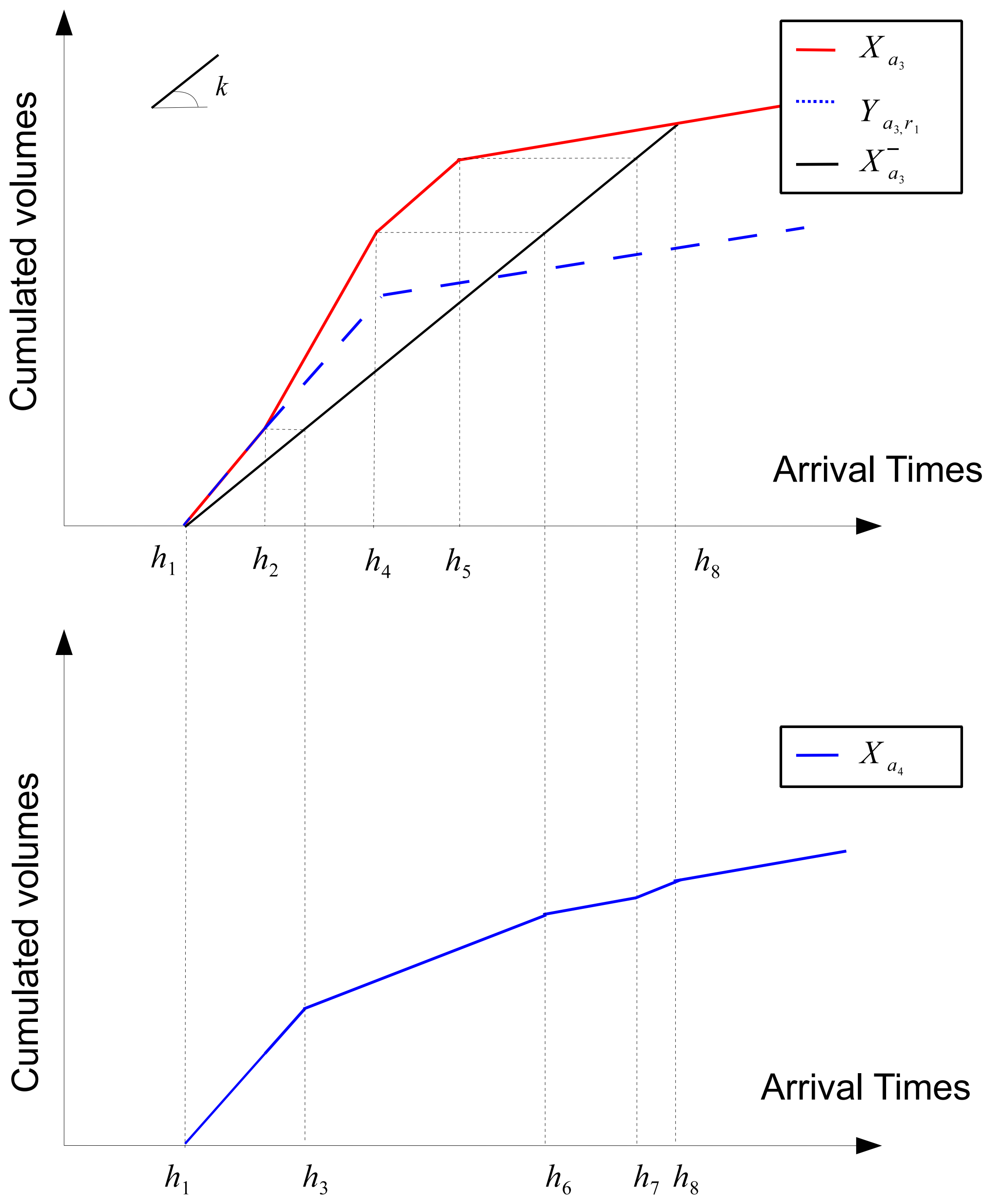

Figure 2 – Dynamic Network Loading solutions

GENERAL ALGORITHM STATEMENT

DATA STRUCTURES

First let us define the proper data structure to model the problem under a discrete event system. Basically, we need to be able to describe a state of the system, to formalize the concept of events and to treat events.

Instantaneous flow vectors and events. A flow vector is a vector of instantaneous flows $(y_r)_{r\in R}$ and has the following physical interpretation: it represents the superposition of the flow of users following different route in a given point at a given instant. The sequence of flow vectors $S^{ext} = (\boldsymbol{y}_a)_{a\in A}$ is called the external state of the system, and represents flows, decomposed according to the followed route, at the entrance of each arc of the network. The term external refers to the fact that this description only focuses on arc incoming flows and totally ignores what's happening inside the links, which we will later refer to as the internal state of a system. Despite that, its knowledge over the simulation period (i.e. for every instants) is exactly what's required to solve the DNLP. Extending $S^{ext} = (\boldsymbol{y}_a)_{a\in A}$ description over the whole period of simulation is hence of interest and so we introduce the concept of event as a change of vector flow at a specific instant and on a specific node.

Formally, an event is defined as:
Definition 1. [Event] An event is a pair $(h, \boldsymbol{e})$ where:
- h is a clock time
- $\boldsymbol{e}$ is a map from $A \times R$ such that $\boldsymbol{e}(a,r)$ is either an instantaneous flow or the empty set $\emptyset$.

Arc event functions. Let's now focus on the arc description. At this point of our exposition, we remain general and only expose how to represent travel time model in an event based perspective. First, one needs to be able to describe the state of an arc. Physically the state of an arc describes at an instant h the traffic flows over the whole arc. In other terms, it is what you cannot see by solely looking at the arc incoming and outcoming flows. In a computational perspective, we would like the knowledge of the state of an arc to be enough to compute all of the future values of the arc travel time and outgoing flows over an infinite time horizon, assuming the route flow vector $\boldsymbol{y}$ at entrance remains constant. According to the causality assumption, in the general case $X_a|_h$ is enough to compute the travel time at h. Consider the route flow vector $\boldsymbol{Y}_a{}'$ obtained by considering for each route $Y_{a,r}\big|_h$ and prolonging it using the instantaneous flow following the corresponding route y_r. The knowledge of all the route volumes thus obtained is then sufficient to compute the travel time for any instants assuming that the incoming flow remains constant. This discussion leads to choose to represent the state of an arc simply by a route volume vector $\boldsymbol{X}$. The inner state of the system is then naturally defined as a collection of route volume vectors $S^{in} = (\boldsymbol{Y_a})_{a\in A}$, one for each arc of the network.

We denote the operation of prolonging a cumulated volume X by an instantaneous flow x $X \oplus_h x$. Note that in PWL format, denoting $X = (h_i, X_i, x_i)_{i=1..n}$, it is equivalent to take the sequence of pieces (h_i, X_i, x_i) such that $h_i < h$ and to add the piece $(h, X(h), x)$.

Definition 2. [Event functions] For each arc travel time model t_a, the following functions are defined:
1. **The next event function** $F_a\colon (\boldsymbol{Y}, h) \to (h', \boldsymbol{e})$, where $\boldsymbol{e}$ is the event representing the first change in slope in the outcoming route volumes $Y_r{}^{-} := Y_r \circ H_a[Y]$ after h. Denote

h' the instant when this change occurs. Then for all $r: a \in r$, consider a' the first arc after a in r and define $\boldsymbol{e}(a', r) := x^{-}_{a,r}$. If h' doesn't exists, then $h' = +\infty$ and $\boldsymbol{e} = \emptyset$.

2. **The handling function** $U_a: \left(\boldsymbol{Y^1}, \boldsymbol{y}, (h, \boldsymbol{e})\right) \rightarrow \boldsymbol{Y}^2$ such that $Y^2_r := Y^1_r \oplus_h \boldsymbol{e}(a, r)$ **if** $\boldsymbol{e}(a, r) \neq \emptyset$ and $Y^2_r := Y^1_r$ otherwise.

Note that the expression $(+\infty, \emptyset)$ encodes the *null event* that indicates that no event is generated by a function F_a

For specific travel time models, such as bottleneck ones, using a route vector flow to describe the sate of an arc might not be the most suitable choice. In this situation the event functions will need to be adapted to feet this new model. Yet the global framework of the algorithm will remain the same. This will be discussed later in the application on the networks of bottlenecks.

ALGORITHM STATEMENT

Now the basic concepts have been settled the reader should begin to see the global picture. At an instant h, we are now able to describe the system by an external state $S^{ext} = (\boldsymbol{y_a})_{a \in A}$ and an internal state $S^{in} = (\boldsymbol{Y_a})_{a \in A}$, representing respectively the traffic flow between the arcs and the traffic flows inside the arcs. Assuming the incoming flow on an arc remains constant, the next event function of this arc enables us to deduce the next change in its outgoing flow. Yet there are two reasons that might cause a change in the instantaneous entrance flow of an arc: the instantaneous ouitgoing flow of another arc changes or the flow leaving an origin changes. The first case is easy to handle: for a given state (S^{in}, S^{ext}) compute the next event for all arcs and only consider the first one chronologically. The second type of events can be treated by a trick: the route volume vectors describing cumulated volumes at origins can always be interpreted as a sequence of events, each of them indicating the change at a given instant of the route flows from a value to another. Combining this set of events, the events at origin, with the previous ones, the arc events, allows determining which will be the next event.

The precise statement of our event-based loading algorithm is given below. The inputs are simply the arcs described by their event functions and the route volumes at origin described by a collection of events. The outputs are route volume vectors, one for each arc, representing the cumulated volumes at the entrance. The algorithm can be summarized as such. Consider the list of event formed by the merge of the events at origins and the events on arcs and remove the first event. Then use the handling function to update the state of the arc concerned with the event. Finally compute the new arc event list using the next event functions of each arc of the network.

Algorithm : `loadingTraffic`

Inputs: - A list of arc $A = (a_1, \dots, a_n)$ together with the corresponding functions H_a and F_a for each $a \in A$.

```
              - A list of events $E^O = (h_1, \boldsymbol{e}_1), \dots, (h_i, \boldsymbol{e}_i), \dots$
Outputs :     - A collection of route flow vector $\boldsymbol{Y}_a$
Initialize:   $\boldsymbol{y}_a$ and $\boldsymbol{Y}_a$ to 0 for all $a \in A$, $E^A$ to the empty list and $h$ to a suitable
initial instant.

While $E^o \cup E^A \neq \emptyset$
    Get next event from $E^o \cup E^A$ and Set it to $(h, \boldsymbol{e})$.
    Forall $(a, r)$: $(\boldsymbol{e}(a, r) \neq \emptyset)$, set $y_{a,r} := \boldsymbol{e}(a, r)$
    Foreach arc $a$ : $\exists r$ :$(\boldsymbol{e}(a, r) \neq \emptyset)$,
        Set $\boldsymbol{Y}_a := U_a(\boldsymbol{Y}_a, (h, e))$
        If $F_a(\boldsymbol{Y}_a, (h, e)) \neq (+\infty, \emptyset)$ , add it to $E^A$
End While
```

APPLICATION TO A NETWORK OF BOTTLENECKS

General presentation

In this section we consider a network of bottlenecks where each arc a is described by two parameters: its free flow travel time t_0^a and its exit capacity k_a. We are going to apply to this network our general algorithm. To do so divide each arc in two parts: the free flow part and the bottleneck part. It is this new network we are going to consider and thus two types of arc time models and event functions have to be defined.

The treatment of the free flow part is straightforward and can be achieved by directly applying the general method presented above. Concerning the bottleneck part, a modification to the next event function is presented below and allows accelerating computation by storing slightly more information in the bottleneck state.

Bottleneck part. As precised earlier, in this case using solely a route flow vector $\boldsymbol{Y}$ to describe the state of a bottleneck is not the best choice from a computational perspective. A more suitable choice is to have some information about the queue evolution. To do so let us add to the state of a bottleneck the function Q. The quantity Q is a positive function of the time representing the queue volume with respect to the time as defined in Equation (5).

It is now necessary to adapt the event functions in order to exploit the additional information given by Q. Given an bottleneck state $(\boldsymbol{Y}, Q)$, how can one compute the next event? Assume the queue is not empty at an instant h. From the analytics exposed above, two cases can arise:

1. The outgoing flow change because of a previous change in the incoming flow. Denoting the corresponding incoming route flows y_r, the new outgoing flows are $y_r^- := \frac{y_r}{\sum_{r' \in R} y_{r'}} k_a$. Finding this instant corresponding to the change in the incoming

flow boils down to find an instant h' such that $h' + \frac{Q(h)}{k_{\boldsymbol{a}}} < h$ and the derivative of a route volume Y_r changes in h'.

2. The outgoing flows change because the queue vanishes. The new outgoing flows are simply the incoming flows $y_r^- := y_r$. Finding the instant h' occurs is straightforward knowing Q.

The event functions for bottleneck are precisely defined below.

Definition 3. [Event functions for bottlenecks] Denote $\boldsymbol{y}_a = \left(y_{a,r}\right)_{r\in R}$the incoming route flow on arc a and $n(a,r)$ the first arc after a in route r. The events functions of a bottleneck are then defined as follow.

1. **The next event function** $F_a: (\boldsymbol{Y}, Q, h) \to (h', e)$.
   - **Case 1:** $Q(h) = 0$
     - If $y_{a,r} = \frac{dY_{a,r}}{dh}$ then $h' := +\infty$ and $\boldsymbol{e} := \emptyset$.
     - else $h' := h$ and for all r define $\boldsymbol{e}(n(a,r), r) := y_{a,r}$.
   - **Case 2:** $Q(h) \neq 0$
     - Let h' the last change in slope of $X_{a,r}$ of such that $h' + \frac{Q(h)}{k_{\boldsymbol{a}}} < h$. Then for any $r \in R$, let x_r be the right derivative of $X_{a,r}$ in h' and define $\boldsymbol{e}(n(a,r), r) := \frac{x_r}{\sum_{r'\in R} x_{r'}} k_a$.
     - If there is no such h', let h' be the first instant such that $Q(h') = 0$. Then for any $r \in R$, let x_r be the right derivative of $X_{a,r}$ in h' and define $\boldsymbol{e}(n(a,r), r) := x_r$.

2. **The handling function** $H_a: \left(\boldsymbol{Y^1}, Q^1, (h, \boldsymbol{e})\right) \to \boldsymbol{Y}^2, Q^2$ such that $Y_{a,r}^2 := Y_{a,r}^2 \oplus_h \boldsymbol{e}(a,r)$. and $Q^2 = Q^1 \oplus_h (\sum_{r\in R} \frac{dY_r^2}{dh} - k_a)$

We claimed that this latter implementation of the event function is more efficient than the former one. Why is that? The general implementation proposed to seek the next event by computing for each call of the function next event the travel time for the current state of the arc. Consequently it does not use at all the information gathered through the previous calls of this function. Yet for a bottleneck model this requires the integration of a first order differential equation of PWL functions and thus a full list traverse. On the contrary, the adaptation for the bottleneck model exploits this information by keeping in memory the queue. The most computational intensive operation is to perform the operation described by the first item of case 2 in Definition 3. Although in the worst case this operation also requires a list traverse, it tends to be a simple scan forward on the last pieces of Q.

Numerical illustration

In this subsection, a small but instructive example is presented. We consider the network of four arcs and two OD pairs presented in Figure 1. Only the arcs $O_1 - D_2$ (arc 1) and $O_2 - D_1$ (arc 2) are subject to bottleneck congestion and they both have a capacity of $k = 2000$ pcu/h. All the arcs have a free flow travel time of $t_0 = 1$. Only two routes are considered, the first one being $O_1 - D_2 - O_2 - D_1$ (route r_1) and the second one $O_2 - D_1 - O_1 - D_2$ (route r_2). The simulation period is $I = [5:00,20:00]$ and the instantaneous flow on r_1 and r_2 are respectively a discretized gaussian shaped curve centered in 10 and a simple constant flow of 1000 pcu/h. The inputs are plotted in Figure 2.

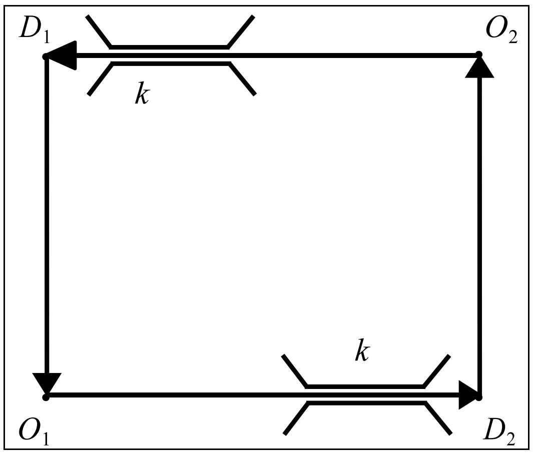

Figure 1 - Network for the numerical illustration

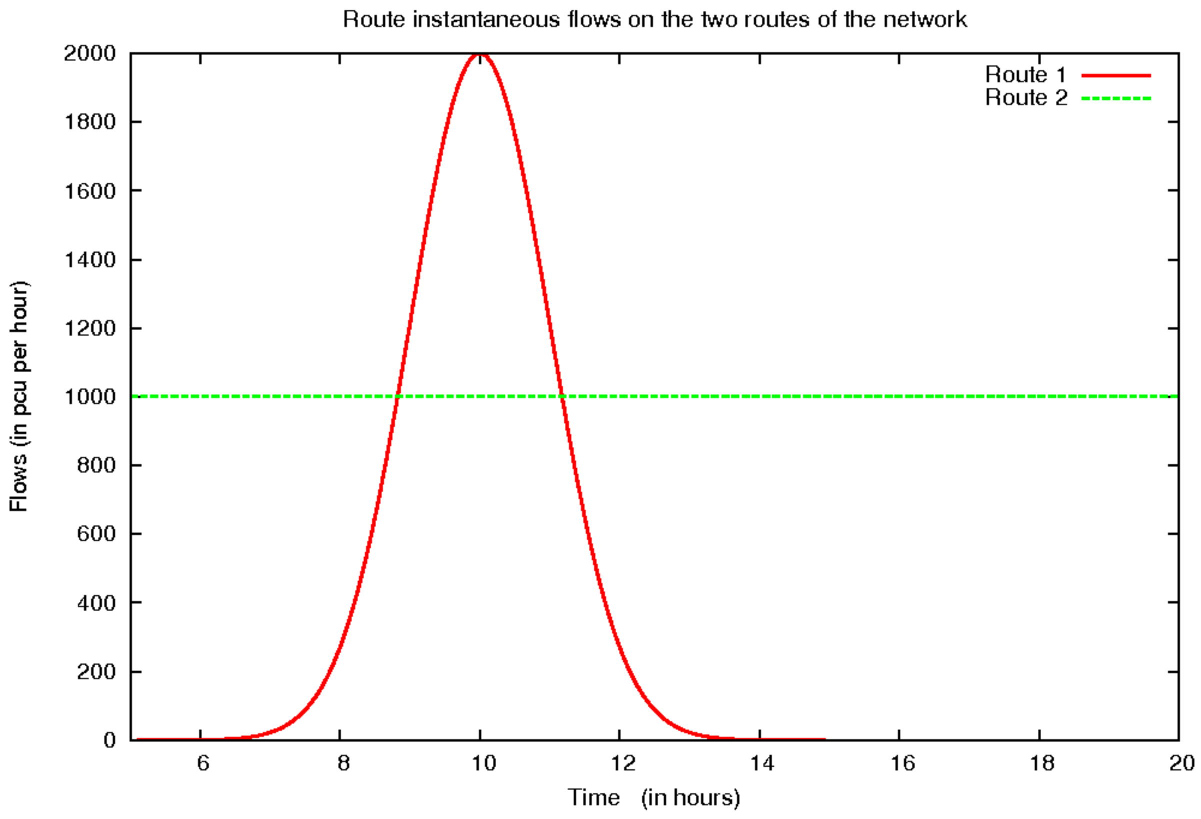

Figure 2 - Inputs

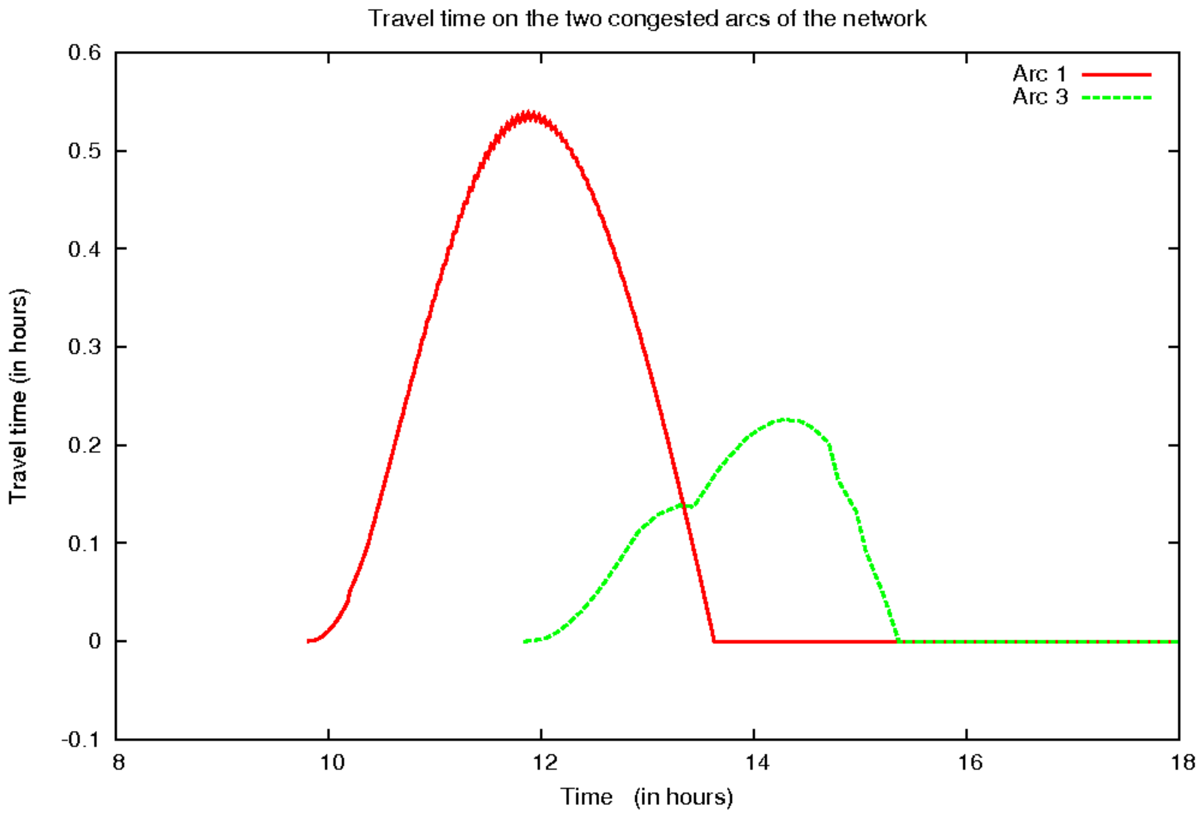

Figure 3 - Numerical results

This network configuration is especially interesting and complex from an event based perspective. Assume that the instantaneous flow on route r_1 increases then the travel time on a_1 is going to increase and eventually the proportion of flows exiting a_1 and following route r_1 will also. This in turn results in a rise in y_{a_3,r_1} and finally in a decrease in y_{a_4,r_2} and in the incoming flow on a_1. The network thus acts as a sort of feedback loop, inducing a decrease in the flow on a route when the flow on the other route grows.

The travel times resulting from the loading of the traffic are plotted in Figure 3. The feedback effect exposed a few line above can be seen on the travel time on a_1. It results in oscillations around the maxima of in travel time. Also note the shape of the travel time on a_3 where two distinct maxima appear.

Benchmark

The running time of the event based algorithm applied on a network of bottlenecks is clearly proportional to the number of events treated. Yet this latter quantity is impossible to compute *a priori*. In this subsection a small numerical experiment is conducted in order to get some insights about the sensitivity of the number of events with respect to the volumes at origin.

The setting is the following. A randomly generated network of 80 nodes and 200 arcs is considered. Then routes of 10 arcs are randomly generated, each of them assigned with a Gaussian shaped flows discretized in 20 pieces. This experiment has been conducted several times with a number of routes varying between 5 and 80.

Why is this numerical experiment relevant? One of the main applications of the DNLP is its integration in dynamic traffic assignment algorithms. Yet in most numerical schemes for dynamic traffic assignment, on progressively discover new routes to serve an origin destination and assign part of the traffic on them. Thus the further the algorithm goes, the more routes are loaded with traffic. Another alternative would have been to try networks of different size. But in the DNLP, the size of the network, on the contrary to many other graph-based algorithms has little influence. The dimensioning quantity is rather the number of routes and the way they overlap themselves.

Figure 5 shows the evolution of the number of events with the number of routes. At first the evolution is roughly linear. Around 50 routes the slopes quickly switch to a much higher value. Around 70 routes it seems that evolution becomes linear again.

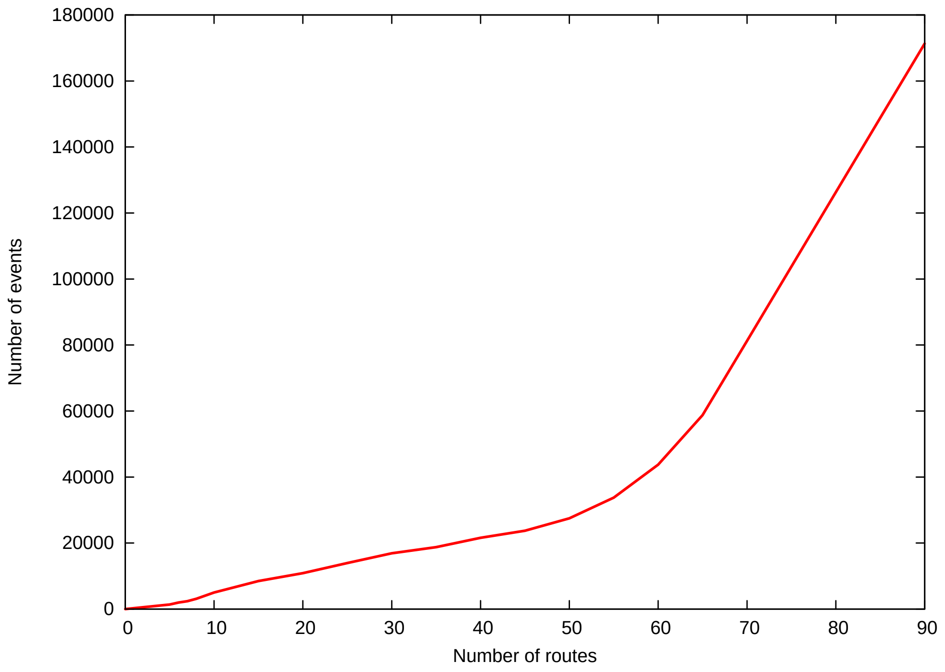

Figure 5 – Number of events variation with the number of assigned routes

A possible explanation for this behaviour is given by the way the events propagate themselves over the network. When the number of routes assigned with traffic is low on a network, those routes tends not to intersect. Consequently the number of event is roughly the number of pieces of the volumes at origin times the average number of arcs on a route. However as the number of routes increases, more routes intersect and quickly an event occurring at given place of the network tends to propagate all over the network. This phenomenon is depicted on Figure 6.

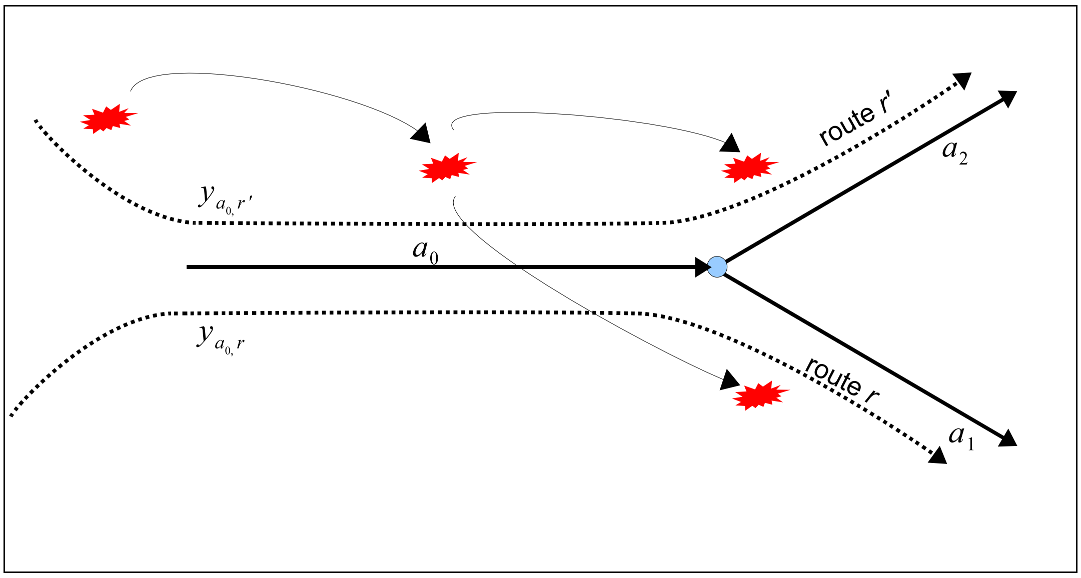

Figure 6 – Event Propagation.
In this simple network, two routes are going through a_0. If an event occurs on route r', upstream arc a_0, it will cause an event on arc a_0 and thus on all downstream arcs of route r.

Note that the number of events is a good proxy for the running time of our algorithm, but also of the actual "complexity" of the result of the dynamic loading. Indeed the number of event treated is essentially the number of pieces of the resulting cumulated flows on the arcs. In that perspective our results can be interpreted in two ways. On one hand, it is good news, as the running time seems to be asymptotically linear with the size of the route flow vector in input. But on the other hand, the practical number of events to deal with a reasonably small example is quite high. For real size networks with an important number of an origin-destination pairs, an efficient exact algorithm seems to be difficult. This is a strong argument for approximate loading procedure.

CONCLUSION

In this paper a generic algorithm for the dynamic network loading problem has been presented and an application to a network of bottleneck has been presented. It was also an opportunity to gives a theoretical insight of the traffic flowing on a network in a dynamic context. Among our findings, we have seen that the outputs of the loading problem quickly grow in complexity due to the complex interaction of the route flows on the networks.

Although the example of application presented here were rather simple, this event based algorithm is an interesting first step toward a much more generic framework for dynamic network loading. The concept of events was here restricted to a change in the incoming flow of an arc. But one could introduce a wide variety of events modelling various physical phenomena. For instance queue spillback could be considered by adding an event type "arc a is full" and updating the upstream arcs in consequence. In the same order of idea dynamic traffic regulation schemes such as dynamic traffic regulation techniques could be modelled in an event based perspective. This offers vast possibilities for future works.

BIBLIOGRAPHY

Daganzo, C. F. (1995). Properties of Link Travel Time Functions under Dynamic Loads, Transportation Research 29B, 95-98.

Friesz, T. L., D. Bernstein, T. E. Smith, R. L. Tobin and B. W. Wie (1993). A Variational Inequality Formulation of the Dynamic Network User Equilibrium Problem Operation Research 41:179-191.

Kuwahara, M. and T. Akamatsu (1993). Dynamic Equilibrium Assignment with Queues for a One-to-Many OD Pattern. Transportation and Traffic Theory, 12, 185-204.

Leurent, F. (2003). On network assignment and supply demand equilibrium: an analysis framework and a simple dynamic model. Proceedings of the European Transport Conference.

Mahmassani, H. S., Hu, T. and R. Jayakrishnan (1995). Dynamic traffic assignment and simulation for advanced network informatics (DYNASMART), Urban traffic networks: Dynamic flow modeling and control. Springer, Berlin/New York.

Meunier, F. and N. Wagner (2010). Equilibrium results for dynamic congestion games. Accpeted for publication in Transportation Science.

Rubio-Ardanaz, J. M., J. H. Wu and M. Florian (2003). Two Improved Numerical Algorithms for the Continuous Dynamic Network Loading Problem. Transportation Research, 37B, 171–190.

Tong, C. O., and S. C. Wong (2000). A predictive dynamic traffic assignment model in congested capacity-constrained road networks, Transportation Research, 34B, 625-644.

Wu, J. H., Y. Chen and M. Florian (1998). The Continuous Dynamic Network Loading Problem: A Mathematical Formulation and Solution Method. Transportation Research 32B, 173–187.

Xu, Y. W., J. H. Wu, M. Florian, P. Marcotte and D. Zhu (1999). Advances in the Continuous Dynamic Network Loading Problem. Transportation Science 33, 341-353.