Shortest Paths and Probabilities on Time-Dependent Graphs - Applications to Transport Networks

Sébastien Felix, Jérôme Galtier

Shortest Paths and Probabilities on Time-Dependent Graphs - Applications to Transport Networks

Sebastien Felix
Orange Labs and Mascotte(INRIA/I3S/CNRS/Univ. de Nice)
905 rue Albert Einstein
06921 Sophia-Antipolis Cedex, France
Email: sebastien.felix@orange-ftgroup.com

Jerome Galtier
Orange Labs
905 rue Albert Einstein
06921 Sophia-Antipolis Cedex, France
Email: jerome.galtier@orange-ftgroup.com

***Abstract*—In this paper, we focus on time-dependent graphs which seem to be a good way to model transport networks. In the first part, we remind some notations and techniques related to time-dependent graphs. In the second one, we introduce new algorithms to take into account the notion of probability related to paths in order to guarantee travelling times with a certain accuracy. We also discuss different probabilistic models and show the links between them.**

I. Introduction

Since the number of vehicles is increasing, it implies an increase of the time spent in congestion. A recent report ([13]) written by the Texas Transportation Institute shows that the average American spent 34 hours stuck in traffic jams in 2009. The cost caused by congestion was estimated at over 100 billion dollars in the United States in 2009. The evolution of this cost is given in Fig. 1.

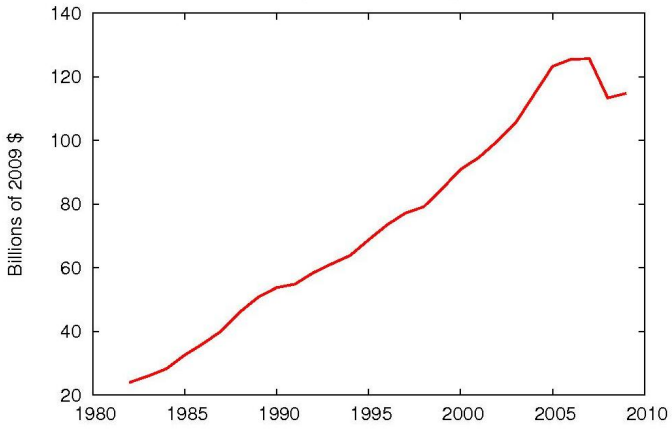


Fig. 1. Total cost of time spent in traffic congestion in the US (data source : [13])

Another observation is that the number of sensors is increasing as well, allowing us to have a better view of the global traffic. It could allow us to have a precise value of the flow on a given road at a given moment (using CCTV cameras, phones, magnetic loops,...), to compute precise Origin/Destination matrices (using GPS on smartphones, particular projection,...) or to determine the travelling time of a path. Since some studies (see [14] and [9]) have shown a link between the load (or the flow) and the speed on a road, then we can obtain information on the travelling time of a path from the flow value on it. But in real-world networks, the flow can be subject to some constraints. As we can see in [2], there exists tools in order to deal with these restrictions.

Another interesting thing is that it is possible to determine the means of transport if the user is provided with an accelerometer and an electronic compass (see [11] and Fig. 2.).

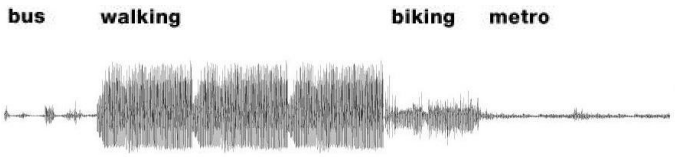


Fig. 2. Accelerometer traces for distinct transportation modes (from http://senseable.mit.edu/co2go/)

We can use this information to establish a better model to manage traffic and optimise it to minimize the time spent in congestion, and then the global cost of transport. Since the evolution of the traffic is dynamic among time, we model it using time-dependent graphs. We also include the notion of probability in this model in order to compute paths whose length is given with a certain accuracy.

This article is organised as follows : in Part II we introduce the notion of time-dependent graphs and show its properties in our specific context. Then in Part III we explain how time-dependent graphs can be exploited to produce itinary results under typical user constraints (given arrival, departure time, or minimizing the delay). In Part IV we focus on the probabilistic issues. We describe an efficient algorithm that can address a large part of these problems but also exhibit more precise - and complex - formulations and explain how to link them with our work and the questions they raise. Finally, in part V we explain how to use the information provided by some sensors in order to have a more precise model of the traffic.

II. Time-Dependent Graphs

When we model transport networks, we are first interested in shortest paths in terms of distance, time or cost. The fact is that the travelling time could vary over time. In order to model these networks, we use time-dependent graphs. A time-dependent graph is given by a directed graph G and a cost function $c(e,t)$. We have $|V(G)| = n$ and $|E(G)| = m$. For every edge $e \in E(G)$, we note $c(e,t) = c_e(t)$. The travelling time of $e = (u,v)$ at the moment t is given by $c_e(t)$. Note that time-dependent graphs were already heavily studied, see [1], [6] and [12].

In time-dependent graphs, we can define the First-In-First-Out (FIFO) property as follow : A time-dependent graph has the FIFO property if and only if $t_1 < t_2 \Rightarrow t_1 + c_e(t_1) < t_2 + c_e(t_2)$, for every $e \in E(G)$. This property implies that $\frac{c_e(t_2)-c_e(t_1)}{t_2-t_1} > -1$. We can define the weak FIFO property on time-dependent graphs by the condition $t_1 < t_2 \Rightarrow t_1 + c_e(t_1) \leq t_2 + c_e(t_2)$ for every $e \in E(G)$. In this case we have $\frac{c_e(t_2)-c_e(t_1)}{t_2-t_1} \geq -1$. Note that the FIFO property implies the weak FIFO property.

When we model transport networks, we have to notice that if two users drive along $e = (u,v)$ they could arrive at the same time in v, although they did not arrive at the same time in u. This could happens if one user waited in u the next planned vehicle in which the other user is, or if one user was stuck in congestion and was later joined by the other user. So we will model transport networks using time-dependent graphs with the weak FIFO property and store the c_e functions using a discrete structure. Moreover for public transport networks, we will use piecewise affine functions of slope -1 as cost functions. We can find more details on time-dependent graphs with the FIFO property in [4].

Let $f_e(t) := t + c_e(t)$ for all $e = (u,v) \in E(G)$. If we leave u at the moment t, we will arrive in v at the moment $f_e(t)$ at the latest. If c_e has the weak FIFO property, we have for all $(x,y) \in \mathbf{R}^2$:

$$\begin{aligned} \frac{c_e(x)-c_e(y)}{x-y} \geq -1 \Rightarrow & \quad \frac{c_e(x)-c_e(y)}{x-y} + \frac{x-y}{x-y} \geq 0 \\ \Rightarrow & \quad \frac{(x+c_e(x))-(y+c_e(y)}{x-y} \geq 0 \\ \Rightarrow & \quad \frac{f_e(x)-f_e(y)}{x-y} \geq 0 \end{aligned}$$

We conclude that f_e is non-decreasing. So if c_e has the weak FIFO property, we can define $f_e^*(t) = \max\{z \in \mathbf{R} / f_e(z) \leq t\}$ (we also have $f_e(t) = \min\{z \in \mathbf{R} / f_e^*(z) \geq t\}$). If we want to be in v at the moment t, we will have to leave u at the moment $f_e^*(t)$ at the latest.

III. Shortest Paths in Time-Dependent Graphs

A. Single Source Shortest Paths

Computing the shortest path from one source s to another vertex v takes almost as much time as computing the shortest paths from s to all other vertices, see [10] and [5]. As we can see in [3], we can adapt Bellman-Ford's and Disjkstra's algorithms to compute the shortest paths from a vertex s to all other vertices in time-dependent graphs. These algorithms build the tree of the shortest paths from s to all other vertices. It is possible to speedup a little the algorithms if we already know a part of this tree.

We show here an adaptation of these algorithms; they take as input the time-dependent graph (G,c), one vertex s and a partial shortest path tree T rooted at s (we can take $T = \{s\}$ if no information is known). Their output are data that allow us to rebuild the shortest paths tree T_s rooted at s which contains all the shortest paths from s to all other vertices starting at the moment t_0.

For each vertex v, we aim to obtain the length d_v of the shortest $s - v$ path starting from s at the moment t_0. This path is composed of the shortest $s - p_v$ path together with the edge (p_v, v). In fact p_v is the predecessor of v in the tree T. Let n_v be the set of successors of v in T. For every $v \in V(G)$, we have $v \in n_{p_v}$. Finally, l_v is the arrival time in v if we use the shortest $s-v$ path starting at the moment t_0.

We present here an algorithm which takes as input a partial shortest paths tree T rooted at s and a time t_0, and computes d_v, l_v, p_v and n_v for every $v \in V(T)$.

Algorithm 1: Tree rebuilding algorithm

Data: A time-dependent graph (G,c), a moment t_0, a vertex $s \in V(G)$ and a partial shortest paths tree T rooted at s.

Result: Shortest paths starting at the moment t_0 from s to all $v \in V(T)$ and their length. We get the outputs d_v, l_v, p_v and n_v for all $v \in V(T)$. d_v is the length of the $s - v$ path in T starting at the moment t_0, which consists in the $s - p_v$ path together with the edge (p_v, v).

Set $l_s = t_0$.
explore(s,T)

The procedure explore(v,T) is defined as follow :

Algorithm 2: explore(v,T)

for *each* $e = (v,w) \in E(T)$ **do**
  $p_w = v$
  $n_v = n_v \cup \{w\}$
  $l_w = l_v + c_e(l_v)$
  $d_w = l_w - t_0$
  explore(w,T)
end

Now we describe an adaptation of Dijkstra's algorithm for time-dependent graphs :

Algorithm 3: Dijkstra algorithm (Shortest paths from one source)

Data: A time-dependent graph (G, c), a moment t_0, a vertex $s \in V(G)$ and a partial shortest paths tree T rooted at s.

Result: Same output as Algorithm 1. We can notice that for every $w \in n_v$ we have $p_w = v$. If v is not reachable from s starting at the moment t_0, then $l_v = \infty$ and $p_v = n_v = \emptyset$.

For each $v \in V(T)$, compute d_v, l_v, p_v and n_v.
Set $l_s = t_0$ and $l_v = \infty$ for all $v \in V(G) \setminus \{s\}$.
Set $R = V(T)$.
while $R \neq V(G)$ **do**
    Find $v \in V(G) \setminus R$ such that $l_v = \min_{w \in V(G) \setminus R}(l_w)$
    Set $R = R \cup \{v\}$.
    for *all* $w \in V(G) \setminus R$ *such that* $e = (v, w) \in E(G)$ **do**
        if $f_e(l_v) < l_w$ **then**
            $l_w = f_e(l_v)$
            $n_{p_w} = n_{p_w} \setminus \{w\}$
            $p_w = \{v\}$
            $n_v = n_v \cup \{w\}$
        end
    end
end
for *each* $v \notin V(T)$ **do**
    $d_v = l_v - t_0$
end

Proposition III.1. *The Dijkstra's algorithm for time-dependent graphs works in* $O(m + n\ln(n))$.

Proof: Obviously, the complexity of this algorithm is the same as the complexity of Dijkstra's algorithm for static graphs. ∎

1) Single Source Shortest Paths in a Time Window: If $(e_1, e_2, ..., e_{k-1}, e_k)$ is a shortest $s - v$ path starting at the moment t, we have $l_v(t) = f_{e_k} \circ f_{e_{k-1}} \circ ... \circ f_{e_2} \circ f_{e_1}(t)$. Since f_e is non-decreasing for every $e \in E(G)$, we can notice that l_v is non-decreasing as well. So if c_e has the weak FIFO property, we can define $l_v^*(t) = \max\{z \in \mathbf{R} / l_v(z) \leq t\}$. If we want to be in v at the moment t, we will have to leave s at the moment $l_v^*(t)$ at the latest.

Let $v \in V(G) \setminus \{s\}$, to find the shortest path from s to v starting at the moment t, we have to compute the sequence $(v_n)_{n \in \mathbf{N}}$ where $v_0 = v$ and $v_i = p_{v_{i-1}}$ for $i \geq 1$. For some i_0 we have $v_{i_0} = s$, then the path $(v_{i_0}, ..., v_1, v_0)$ is the shortest $s - v$ path starting at the moment t.

Of course we can run this algorithm for many values of t_0. Then we could get the functions $l_v(t)$, $d_v(t)$, $p_v(t)$ and $n_v(t)$ defined like in the previous algorithm and related to the departure time t. We have to notice that if every $c_e(t)$ has the weak FIFO property, then the function $d_v(t)$ has also the weak FIFO property. Since we cannot run this algorithm for every value t in an interval I, so we have to compute it for a discrete set of values. Let $I = [t_1, t_2]$ be an interval in $\mathbf{R}$, and $\epsilon \in \mathbf{R}^+$ such that $\frac{t_2 - t_1}{\epsilon} \in \mathbf{N}$. We set $I_\epsilon = \left\{t_1 + k\epsilon : k \in \left\{0, ..., \frac{t_2 - t_1}{\epsilon}\right\}\right\}$.

Proposition III.2. *If* $f(t)$ *has the weak FIFO property, then we have* $\min_{t \in I}(f(t)) \geq \min_{t \in I_\epsilon}(f(t)) - \epsilon$.

Proof: Let $t_m \in I$ such that $f(t_m) = \min_{t \in I}(f(t))$. If $t_m \in I_\epsilon$, the inequality holds. If $t_m \notin I_\epsilon$, let $t_0 \in I_\epsilon$ such that $t_0 = \max(t \in I_\epsilon / t < t_m)$. We have $0 < t_m - t_0 < \epsilon \Leftrightarrow \frac{1}{t_m - t_0} > \frac{1}{\epsilon}$.

If $\min_{t \in I}(f(t)) < \min_{t \in I_\epsilon}(f(t)) - \epsilon$, we have :
$f(t_m) < f(t_0) - \epsilon \Leftrightarrow f(t_0) - f(t_m) > \epsilon$.

If we combine these two inequalities, we obtain :
$\frac{f(t_0) - f(t_m)}{t_m - t_0} > 1 \Leftrightarrow \frac{f(t_m) - f(t_0)}{t_m - t_0} < -1$.

So $f(t)$ does not have the FIFO property, which is a contradiction. Then $\min_{t \in I}(f(t)) \geq \min_{t \in I_\epsilon}(f(t)) - \epsilon$. ∎

So if we set $\epsilon > 0$, we can run this algorithm and compute $d_v(t)$ for $t = t_1 + k\epsilon$ where $0 \leq k \leq \frac{t_2 - t_1}{\epsilon}$. Since $d_v(t)$ has the weak FIFO property, we have $\min_{t \in I}(d_v(t)) \geq \min_{t \in I_\epsilon}(d_v(t)) - \epsilon$. So there will be a maximum uncertainty of ϵ between the minimum and the computed value.

If we had chosen c_e with the weak FIFO property for some $e \in E(G)$, the value $r_v(t_0) := \min_{t \geq t_0}(l_v(t))$ would be the moment when we could reach v at the earliest starting from s at a moment $t \geq t_0$. Since in our study l_v has the weak FIFO property and hence is increasing, the minimum is of course attained for $t = t_0$.

2) Single Destination Shortest Paths: There is of course some kind of dual problem, which is to find for every $t \in \mathbf{R}$ a shortest path from every $v \in V(G) \setminus \{s\}$ to s arriving at the moment t. It is possible to modify the previous algorithms in order to solve that problem. We just have to use f_e^* functions instead of f_e functions, replace each edge (u, v) in G by (v, u), replace $<$ by $>$ and min by max in the previous algorithms. The value $l_v(t)$ will be the latest moment when we need to leave v in order to arrive in s before t.

B. Updating the Length of an Edge

In a transport network, we know that it is possible to predict the flow on an edge if we know Origin-Destination matrix over time and the flow on the whole network at a given time. But there still exists impredictable events that

can increase the travelling time of an edge (accidents or malfunction of some equipment). If such an event happens on an edge we were going to cross, it would then be useful to recompute a shortest path to our destination knowing how the event will affect the traveling time.

We can easily take benefit from the shortest paths tree obtained during the algorithm used to compute our desired path. Let $\mathcal{C}_{e,v}(T)$ be the connected component of $T \setminus e$ that contains v. Suppose that we have run a shortest path algorithm to find shortests path from s to all other vertices during the interval $I = [t_1, t_2]$. Let $T_{s,t}$ be the tree that contains shortest paths starting from s at the moment t to all other vertices. We have computed $T_{s,t}$ for every $t \in I$. If an event happens on an edge $e = (u, v)$ and modifies the travelling time from δ_1 to δ_2, it will modify the shortest paths trees $T_{s,t}$ for every t from $l_u^*(\delta_1)$ to $l_u^*(\delta_2)$ (due to the definition of l_u^*). We finally just have to use one of the previous algorithms to build the new shortest paths trees for $t \in [l_u^*(\delta_1), l_u^*(\delta_2)]$ knowing that $\mathcal{C}_{e,u}(T_{s,t})$ is a partial shortest paths tree at the moment t. Since we just have to compute for each $t \in [l_u^*(\delta_1), l_u^*(\delta_2)]$ the data for the vertices $w \in \mathcal{C}_{e,v}(T_{s,t})$, the execution of this algorithm is faster than a standard single source shortest path algorithm though they have the same time complexity.

Proposition III.3. *Let X be a discrete set of departure times from s. Rebuilding the shortest paths tree after an update of an edge's length which affects the shortest paths from s starting at t for $t \in X$ can be done in $O(|X|\,(m + n\ln(n)))$.*

Proof: For each $t \in X$, we will have to run at worst a full Dijkstra's algorithm. So the complexity of this algorithm is $O(|X|\,(m + n\ln(n)))$.

■

Of course this idea could be adapted in the case where we look for shortest paths from all vertices to a single vertex s with very few changes.

IV. Single Source Shortest Path including Probabilities

In practice, we cannot guarantee an exact travelling time. We could give such a time but only with a certain probability. Some work has already been done on graphs with random edge's lengths, see [7] and [8]. Let G be a digraph. For each edge $e = (u, v)$, we have the function $\delta_e : \mathbf{R}\times\mathbf{R} \to [0, 1]$ such that $\delta(t_d, \cdot)$ is the probability density function of the random variable that gives the duration of traversing e starting at t_d. In other words, the probability of traversing e in duration t_a starting at t_d is

$$\int_0^{t_a} \delta(t_d, z)dz.$$

. Let $D(G) = \{\delta_e / e \in E(G)\}$, we say that $(G, D(G))$ is a time-dependent graph with probabilistic weights. If $p_e(t_1, t_p) = p_e(t_2, t_p)$ for every $(t_1, t_2, t_p) \in \mathbf{R}^3$ and $e \in E(G)$, we say that $(G, D(G))$ is a static graph with probabilistic weights.

We have $\int_{-\infty}^{+\infty} \delta_e(t_d, \epsilon)\, d\epsilon = 1$, $\forall (e, t_d) \in E(G) \times \mathbf{R}$. If $e = (u, v)$, let $\Delta_e^-(t_d) = \inf\{t \in \mathbf{R} / \delta_e(t_d, t) > 0\}$ (respectively $\Delta_e^+(t_d) = \sup\{t \in \mathbf{R} / \delta_e(t_d, t) > 0\}$) be the earliest moment (respectively the latest moment) when we could reach v if we have left u at the moment t_d. We set $p_e(t_d, t_p) = \int_{-\infty}^{t_p} \delta_e(t_d, \epsilon)\, d\epsilon$, the probability to traverse $e = (u, v)$ in a duration less than or equal to t_p if we start at $t = t_d$. Let $t_d \in \mathbf{R}$, we set :

$$\begin{array}{rccc} \pi_{e,t_d} : & [\Delta_e^-(t_d), \Delta_e^+(t_d)] & \to [0, 1] \\ & t & \mapsto p_e(t_d, t) \end{array}$$

We note that π_{e,t_d} is a bijection, so we can define :

$$\begin{array}{rccc} c_e : & \mathbf{R} \times [0, 1] & \to \mathbf{R} \\ & (t_d, p) & \mapsto \pi_{e,t_d}^{-1}(p) \end{array}$$

Let $e, e' \in E(G)$ and $\delta_{e+e'}$ be the density function of travelling times for the path $e + e'$. We can compute $\delta_{e+e'}(t_d, t_p)$, which is probability density to traverse $e + e'$ in a time t_p if we started at the moment t_d. If e is traversed in a time z, the probability density associated is $\delta_e(t_d, z)$. Then e' needs to be traversed in a time $t_p - z$ starting at time $t_d + z$ and the probability density associated is $\delta_{e'}(t_d + z, t_p - z)$. If we integrate the product of these probability densities for every value of z, we obtain the formula $\delta_{e+e'}(t_d, t_p) = \int_{z=-\infty}^{+\infty} \delta_e(t_d, z)\delta_{e'}(t_d + z, t_p - z)dz$.

Notice that if the graph is static, the previous formula is exactly the convolution of δ_e and $\delta_{e'}$.

Proposition IV.1. *For every $t_1 \in \mathbf{R}$, we have :*

$$p_{e+e'}(t_d, t_p) \geq p_e(t_d, t_1) * p_{e'}(t_d + t_1, t_p - t_1).$$

Proof: First, note that the FIFO property implies that for all $\delta \geq 0$ and $(t_d, t_p) \in$

Using definitions, we obtain :

$$\begin{array}{rcl} p_{e+e'}(t_d, t_p) = & \int_{t=0}^{t_p} \delta_{e+e'}(t_d, t)dt \\ = & \int_{t=0}^{t_p} \int_{z=-\infty}^{+\infty} \delta_e(t_d, z)\delta_{e'}(t_d + z, t - z)dzdt \\ = & \int_{z=0}^{t_p} \delta_e(t_d, z) \int_{t=z}^{t_p} \delta_{e'}(t_d + z, t - z)dtdz \\ \geq & \int_{z=0}^{t_1} \delta_e(t_d, z) \int_{t=z}^{t_p} \delta_{e'}(t_d + z, t - z)dtdz \end{array}$$

If we set $u = t - z$, we have :
$\int_{t=z}^{t_p} \delta_{e'}(t_d + z, t - z)dt = \int_{u=0}^{t_p - z} \delta_{e'}(t_d + z, u)du$

Then using formula (1) with $\delta = t_1 - z$, we obtain :
$\int_{t=z}^{t_p} \delta_{e'}(t_d + z, t - z)dt \geq \int_{u=0}^{t_p - t_1} \delta_{e'}(t_d + t_1, u)du$

We conclude that :

$$\begin{aligned} p_{e+e'}(t_d, t_p) \geq & \int_{z=0}^{t_1} \delta_e(t_d, z) \int_{u=0}^{t_p - t_1} \delta_{e'}(t_d + t_1, u) du dz \\ \geq & \int_{z=0}^{t_1} \delta_e(t_d, z) dz * \int_{u=0}^{t_p - t_1} \delta_{e'}(t_d + t_1, u) du \\ \geq & \quad p_e(t_d, t_1) * p_{e'}(t_d + t_1, t_p - t_1) \end{aligned}$$

■

We can notice that if we set $t_p = t_1 + t_2$, proposition IV.1 inplies :
$p_{e+e'}(t_d, t_1 + t_2) \geq p_e(t_d, t_1) * p_{e'}(t_d + t_1, t_2)$.

If we start from u at $t = t_d$, we have a probability p to traverse e in a duration less than or equal to $c_e(t_d, p)$. Let $e = (u, v)$ and $e' = (v, w)$ two edges of G. Let p_a be the probability to traverse e_a in a duration less than or equal to t_a, starting from u at time t_d. Thanks to Proposition IV.1, the probability to traverse $e + e'$ in a duration less than or equal to $t_a + c_{e'}(t_d + t_a, p_b)$ is at least $p_a * p_b$. We can then define the length of a path on which we fix the desired probabilities for each edge.

We are now interested in computing the time needed to traverse a path with a given accuracy (i.e. a given probability). Typically, we want to be able to say that a path will be traversed in a time less than or equal to t_p with a probability p (where p has already been fixed).

Let $P = (e_1, ..., e_n)$ be a path in G where $e_i = (v_{i-1}, v_i)$, we note $P_i = (e_1, ..., e_i)$ the sub-path of P containing the i first edges. We can associate the travelling time of P with a probability p_i on e_i for every n-uple $(p_1, p_2, ..., p_n) \in [0, 1]^n$. Let $E = [0, 1]^n$, we note $E_i = [0, 1]^i$. Let $X = (x_1, ..., x_n) \in E$, we note $X_i = (x_1, ..., x_i) \in E_i$.

We can compute the time needed to traverse P_i starting from v_0 at $t = t_d$ with the probabilities X_i, noted $C_{P_i}(t_d, X_i)$, using the recursive formula :
$C_{P_i}(t_d, X_i) = C_{P_{i-1}}(t_d, X_{i-1}) + c_{e_i}(t_d + C_{P_{i-1}}(t_d, X_{i-1}), x_i)$.

We can notice that $C_{P_{i-1}}(t_d, X_{i-1})$ is the time needed to traverse P_{i-1} starting from v_0 at $t = t_d$ with the probabilities X_{i-1} and $c_{e_i}(t_d + C_{P_{i-1}}(t_d, X_{i-1}), x_i)$ is the time needed to traverse e_i starting from v_{i-1} at $t = C_{P_{i-1}}(t_d, X_{i-1})$ with the probability x_i.

Let $H_{P_i, t_d}(p) = \left\{ X_i \in E_i / \prod_{j=1}^{i} x_j = p \right\}$ the subset of E_i that contains all i-uples $(p_1, ...p_i)$ that guarantee a total travelling time with at least a probability p. Our goal is to find the shortest travelling time for P_i with at least a probability p and the corresponding i-uple in $H_{P_i, t_d}(p)$. So we can define $c_{P_i, t_d}(p) = \min_{X_i \in H_{P_i, t_d}(p)} (C_{P_i}(t_d, X_i))$ which corresponds to the shortest travelling time for P_i starting at t_d with at least a probability p.

Of course, it is not realistic to compute the exact value. But we can still find an approximation, computing the minimum of $C_{P_i}(t_d, X_i)$ on a discrete subset of $H_{P_i, t_d}(p)$. Let $\eta = 1 - 10^{-N}$ with $N \in \mathbf{N}^*$. We present here some algorithms that give for a given $s \in V(G)$ the length of all shortest $s - v$ paths with a probability higher than or equals to η^M for $M \in \mathbf{N}^*$. Since we are interested in high probability paths, we can look for the minimum of $C_{P_i}(t_d, X_i)$ for $X_i \in H_{P_i, t_d}(p)$ such that every x_j belongs to $\{\eta^q / 0 \leq q \leq M\}$.

We describe here such an algorithm based on Dijkstra's single source shortest path algorithm :

Algorithm 4: Algorithm for shortest path from one source including probabilities

Data: A digraph G with $c_e(t_d, p)$ for every $e \in E(G)$.
Result: For every $v \in V(G)$ we have an output $(l_v(n))_{0 \leq n \leq M}$, where $l_v(i) = (t_i, \{p(v), t_v\})$. Note that t_i is the time needed to reach v from s with a probability higher than η^i if we start at $t = t_d$, knowing we left the predecessor $p(v)$ of v at time t_v

.

Set $(l_s(n))_{0 \leq n \leq M}$ with $l_s(i) = (t_d, \emptyset)$.
Set $(l_v(n))_{0 \leq n \leq M}$ with $l_v(i) = (\infty, \emptyset)$ for all $v \in V(G) \setminus \{s\}$.
for $i = 0$ **to** M **do**
  $R = \emptyset$
  while $R \neq V(G)$ **do**
    Find $v \in V(G) \setminus R$ such that $l_v(i)[1] = \min_{w \in V(G) \setminus R} (l_w(i)[1])$.
    Set $R = R \cup \{v\}$.
    for *all* $w \in V(G) \setminus R$ *such that* $e = (v, w) \in E(G)$ **do**
      for *every* j *such that* $0 \leq j \leq i$ **do**
        if $l_w(i)[1] > l_v(j)[1] + c_e(l_v(j)[1], \eta^{i-j})$ **then**
          Set $l_w(i) = (l_v(j)[1] + c_e(l_v(j)[1], \eta^{i-j}), \{v, l_v(j)[1]\})$
        end
      end
    end
  end
end

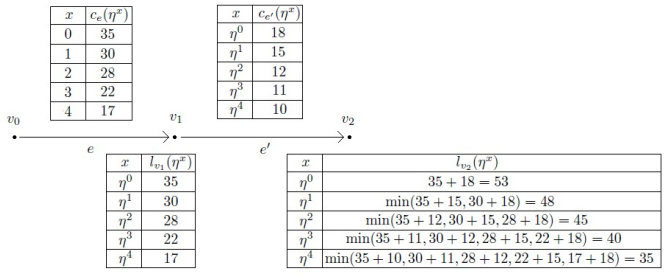
Theorem IV.2. *The algorithm for shortest path from one source including probabilities works in* $O(M^2(m + n \ln(n)))$.

Proof: Each vertex will only be selected once in each loop of the first for. Since in each iteration i of the first loop for we have to do a basic Dijkstra's algorithm with i comparisons,

we will need to do $\frac{M(M-1)}{2}$ comparisons for each vertex. So the complexity of the algorithm for shortest path from one source including probabilities is $O(M^2(m + n\ln(n)))$.

■

This algorithm is based on the fact that if $P = (v_0, v_1, ..., v_n)$ is a shortest $v_0 - v_n$ path whose length is guaranteed with a probability p, then $(v_0, ..., v_i)$ is a shortest $v_0 - v_i$ path whose length is guaranteed with a probability $p'p$ and $(v_i, ..., v_n)$ is a shortest $v_i - v_n$ path whose length is guaranteed with a probability $\frac{p}{p'}$ for every $i = 1, ..., n$. We have $l_{v_i}(t_d, \eta^x) = \min_{j=0,...,x}(l_{v_{i-1}}(t_d, \eta^j) + c_{(v_{i-1},v_i)}(l_{v_{i-1}}(t_d, \eta^j), \eta^{x-j}))$. Note that in the static case we have $l_{v_i}(\eta^x) = \min_{j=0,...,x}(l_{v_{i-1}}(\eta^j) + c_{(v_{i-1},v_i)}(\eta^{x-j}))$. We present here an example of our algorithm applied on a static graph in order to have a clear illustration.



x	$c_e(\eta^x)$
0	35
1	30
2	28
3	22
4	17

x	$c_{e'}(\eta^x)$
η^0	18
η^1	15
η^2	12
η^3	11
η^4	10

x	$l_{v_1}(\eta^x)$
η^0	35
η^1	30
η^2	28
η^3	22
η^4	17

x	$l_{v_2}(\eta^x)$
η^0	$35 + 18 = 53$
η^1	$\min(35 + 15, 30 + 18) = 48$
η^2	$\min(35 + 12, 30 + 15, 28 + 18) = 45$
η^3	$\min(35 + 11, 30 + 12, 28 + 15, 22 + 18) = 40$
η^4	$\min(35 + 10, 30 + 11, 28 + 12, 22 + 15, 17 + 18) = 35$

Fig. 3. An example of the algorithm for shortest path from one source including probabilities

We are now able to compute a higher bound of the time needed to traverse a path with a given probability. We could also be interested in the probability to traverse a path in a time less than or equal to a given time t.

Let $F_{P_i,t_d}(t) = \{X_i \in E_i / C_{P_i}(t_d, X_i) \leq t\}$. In fact the measure of $F_{P_i,t_d}(t)$ gives the probability to traverse the path P_i in a time less than or equal to t_p starting at $t = t_d$. Let $|F_{P_i,t_d}(t)|$ be the measure of $F_{P_i,t_d}(t)$, we have :

$$|F_{P_i,t_d}(t)| = \int_{X_{i-1} \in E_{i-1}} p_{e_i}(t_d + C_{P_{i-1}}(t_d, X_{i-1}), t - C_{P_{i-1}}(t_d, X_{i-1}))dX_{i-1}.$$

If we leave v_0 at $t = t_d$, we will have at least a probability $|F_{P_i,t_d}(t_a - t_d)|$ to arrive in v_i before $t = t_a$.

V. Applications to Transport Networks

We saw in the previous sections that if we have for every $e \in E(G)$ a density function $\delta_e(t_d, t_p)$ such that the probability to traverse e in a time t_p, if we started at $t = t_d$, is $\int_0^{t_p} \delta(t_d, z)dz$. The main question is how to get these δ_e functions.

Since the number of sensors is increasing, we could use the data obtained to determine a lot of information. For example, we can measure the time needed to traverse a path using mobile phones, geolocation techniques or CCTV cameras. We could also use mobile phones, magnetic loops or CCTV cameras to directly estimate the flow value on a road at a given moment, and since there is a function that makes a link between the flow and the speed on a road (see [9] and [14]), we can compute the time needed to traverse a path. Moreover, we can use these sensors to determine O/D matrices and try to predict the flow on a given road at a given moment using multi-commodity flows.

All these techniques could be used to obtain information on the flow, the load and the travelling time of a road and then determine an approximation of the δ_e functions using statistics.

VI. Conclusion and Further Work

We can now choose paths that guarantee a travelling time with a certain probability in time-dependent graphs. In fact, the obtained probability is lower than the real one since we worked with a discrete model and considered the worst case at each step.

We have expressed in this article different formulations that give the probability related to a travelling time on a given path. This opens the way for research work on better approximations of this fonction. We can of course think of an exact computation, but maybe also good approximations will offer a good compromise between accurate computation and real-time needs.

With this, Orange Labs continues to work on the precision of its random models, gathering more information for a better prediction of the traffic. We think that the random models given by this research might have very specific properties that could be exploitable for faster calculations.

In this paper, we have also merged together on purpose "car" traffic problems and "public transport" ones. Anyway, it is clear that these conditions are very different in nature (for instance, the "car" model argues for the strong FIFO property while the "public transport " implies only the weak one), and naturally lead to different mathematical and/or algorithmical issues. Both models are strongly different also on their own impact to traffic jams. On-the-fly adaptation strategies are also very different (rerouting with explosion of the possible options for the first, multimodularity with lots of specific cases, such as acceptation or not of alternative means of transports as bikes, long pedestrian connections, etc...).

We would like to thank J. Chambon for many helpful remarks.

REFERENCES

[1] I. Chabini. Discrete Dynamic Shortest Path Problems In Transportation Applications: Complexity And Algorithms With Optimal Run Time. *Transportation Research Records*, 1645:170–175, 1998.

[2] R. M. Colombo, P. Goatin, and M. Rosini. On the modeling and management of traffic. *Quaderni del Seminario Matematico di Brescia*, 14, 2010.

[3] B. C. Dean. Continuous-Time Dynamic Shortest Path Algorithms, 1999.

[4] B. C. Dean. Shortest Paths in FIFO Time-Dependent Networks: Theory and Algorithms. Technical report, Massachusetts Institute Of Technology, 2004.

[5] E. W. Dijkstra. A Short Introduction to the Art of Programming, 1971.

[6] T. Flatberg, G. Hasle, O. Kloster, E. J. Nilssen, and A. Riise. Dynamic and Stochastic Aspects in Vehicle Routing - A Literature Survey. *STF90 A05413 SINTEF Applied Mathematics*, page 16, 2005.

[7] M. Hua and J. Pei. Probabilistic Path Queries in Road Networks: Traffic Uncertainty Aware Path Selection. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 347–358, New York, NY, USA, 2010. ACM.

[8] M. Jamali. Learning to Solve Stochastic Shortest Path Problems. Technical report, Sharif University of Technology, Tehran, Iran, 2006.

[9] E. Köhler and M. Skutella. Flows over time with load-dependent transit times. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '02, pages 174–183, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[10] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 4th edition, 2007.

[11] V. Manzoni, D. Manilo, K. Kloeckl, and C. Ratti. Transportation mode identification and real-time CO2 emission estimation using smartphones. Technical report, Massachusetts Institute of Technology, 2010.

[12] G. Nannicini. Point-to-Point Shortest Paths on Dynamic Time-Dependent Road Networks. *4OR*, 8(3):327–330, 2010.

[13] D. Schrank, T. Lomax, and S. Turner. TTI's 2010 Urban Mobility Report. Technical report, Texas Transportation Institute, 2010.

[14] Y. Sheffi. *Urban Transportation Networks*. Prentice-Hall, New Jersey, 1985.