ABSTRACT

The deployment of ITS must be assisted by suitable tools to conduct the feasibility studies required for testing the designs and evaluating the expected impacts. Microscopic traffic simulation has proven to be the suitable methodological approach to achieve these goals. This paper discusses one of the most critical aspects of the dynamic simulation of road networks based on a microscopic approach, namely how it performs a heuristic dynamic assignment, the implied route choice models, and whether under certain criteria it can achieve a stochastic user equilibrium.

1. INTRODUCTORY REMARKS

From an analytical point of view dynamic traffic assignment has been usually related to the concept of the dynamic user equilibrium problems. Some of the most successful approaches are inspired on the seminal paper by Friesz et al. 1993, that proposes a dynamic network user equilibrium model which equilibrates the disutilities of the temporal choices. To achieve such equilibrium they take the perspective “that the essential choices available to users of a transportation network – route choice and departure time – occur in time-varying environments and are made rationally”, and they conclude with the assumption that these rational choices can only be made if the disutilities of the alternatives are equilibrated.

Two main approaches have been used to model these route choices. One based on a generalization of Wardrop’s first principle of static traffic assignment, in which users try to optimize their route based on the current information, this approach describes the evolution of flows when users make route choice decisions based on experienced travel times, and it is usually known as a preventive or en-route assignment, it does not achieve a day-to-day equilibrium pattern, therefore it is considered a dynamic traffic assignment principle and not a true equilibrium. In the above referenced paper Friesz et al. propose an alternative generalization of Wardrop’s principle stated in the following terms: If, at each instant in time, for each OD pair, the flow unit costs on utilized paths are identical and equal to the minimum instantaneous unit path cost, the corresponding flow pattern is said to be in dynamic traffic equilibrium. This approach, also known as reactive assignment, can be interpreted in terms that could correspond to users having access to a real-time driver information traffic forecasting system or, alternatively, as an approximation to a process by which traveler combine the experienced travel times with conjectures to forecast the temporal variations in flows and travel costs.

In the above reference the dynamic equilibrium problem is formulated in the space of path flows $h_k(t)$, for all paths $k \in K_i$, the set of feasible paths for the i-th OD pair at time t. The path flow rates in the feasible region $\Omega$ satisfy at any time $t \in (0,T)$ the flow conservation and non-negativity constraints:

$$\Omega = \left\{ h(t) | \sum_{k \in K_i} h_k(t) = g_i(t) ; h_k(t) \geq 0 \right\}$$

for almost all $t \in (0,T)$

where I is the set of all OD pairs in the network, T is the time horizon, and $g_i(t)$ is the fraction of the demand for the i-th OD pair during the time interval t. The approaches assume that the optimal user equilibrium conditions can be defined as (Friesz et al. 1993): A temporal version of the static Wardrop user optimal equilibrium conditions, that can be formulated as:
Where $s_k(t)$ is the path travel time on path $k$ determined by the dynamic network loading. Friesz et al. 1993 show that these conditions are equivalent to the variational inequality problem consisting on finding $h^* \in \Omega$ such that:

$$\left[ S(h^*) h - h^* \right] \geq 0, \forall h \in \Omega$$

According to (Florian et al., 2001), a dynamic traffic assignment model consists of two main components:

1. A method to determining the path dependent flow rates on the paths on the network, and
2. A Dynamic Network Loading method, which determines how these path flows give raise to time-dependent arc volumes, arc travel times and path travel times.

In the Dynamic Network Loading, also known as Dynamic Network Flow Propagation, (Cascetta, 2001), "models simulate how the time-varying continuous path flows propagate through the network inducing time-varying in-flows, out-flows and link occupancies”. A wide variety of approaches, from analytical, (Wu, 1991; Wu et al., 1998a; Wu et al., 1998b; Xu et al., 1998; Xu et al., 1999), to simulation based, (Florian et al., 2001), have been proposed. In all them path flows are determined by an approximate solution to the mathematical model for the dynamic equilibrium conditions. The differences between the various referenced approaches lay in the discretization scheme used to solve (3), and the algorithmic approach to solve the discretized problem (i.e. projection algorithms, successive averages, etc.) and the dynamic network loading mechanism, analytical (Wu, 1991; Wu et al., 1998a; Wu et al., 1998b; Xu et al., 1998; Xu et al., 1999), or simulation based (Florian et al. 2001).

2. HEURISTIC DYNAMIC TRAFFIC ASSIGNMENT BASED ON MICROSCOPIC SIMULATION

The assessment by simulation of ITS applications requires a substantial change in the traditional paradigms of microscopic simulation, in which vehicles are generated at the input sections in the model, and perform turnings at intersections according to probability distributions. In such model vehicles have neither origins nor destinations and move randomly on the network. The required simulation approach must be based on a route based microscopic simulation paradigm. In this approach, vehicles are input into the network according to the demand data defined as an O/D matrix (preferably time dependent) and they drive along the network following specific paths in order to reach their destination. In the route based simulation new routes are to be calculated periodically during the simulation, and a Route Choice model is needed, when alternative routes are available to determine how the trips are assigned to these routes.

The key question that this approach raises is whether this simulation can be interpreted in term of a stochastic heuristic dynamic traffic assignment or not. We propose to investigate the answer to this question in the case of a microscopic simulation using AIMSUN (2003), a route based microscopic simulator (Barceló et al. 1995, 1998). This paper is more elaborated version of a previous research reported in Barceló and Casas 2002 and 2004, and Barceló 2004.

This process can be interpreted in terms of an heuristic approach to dynamic traffic assignment similar to the one proposed by Florian et al. 2001, consisting on:

1. A method to determining the path dependent flow rates on the paths on the network, based on a Route Choice function, and
2. A Dynamic Network Loading method, which determines how these path flows give raise to time-dependent arc volumes, arc travel times and path travel times, heuristically implemented by microscopic simulation.

The implemented simulation process, Barceló and Casas 2004, and Barceló et al. 1995, based on time dependent routes consists of the following procedure, whose conceptual diagram is depicted in Figure 1, is as follows:

$$s_k(t) = \begin{cases} u_i(t) & \text{if } h_k(t) > 0 \\ \geq u_i(t) & \text{Otherwise} \end{cases} \text{ for } \forall k \in K_i, \forall i \in I, \text{for almost all } t \in (0, t) \text{, } h_k(t) \in \Omega$$

(2)
Procedure heuristic dynamic assignment

**Step 0:** Calculate initial shortest path(s) for each O/D pair using the defined initial costs.

**Step 1:** Simulate for a time interval $\Delta t$ assigning to the available path $K_i$ the fraction of the trips between each O/D pair $i$ for that time interval according to the probabilities $P_k$, $k \in K$ estimated by the selected route choice model.

**Step 2:** Update the link cost functions and recalculate shortest paths, with the updated link costs.

**Step 3:** If there are guided vehicles, or variable message panels proposing a rerouting, provide the information calculated in 2 to the drivers that are dynamically allowed to reroute on trip.

**Step 4:**
- **Case a (Preventive dynamic assignment):**
  - If all the demand has been assigned then stop. Otherwise go to step 1.

- **Case b (Reactive dynamic assignment):**
  - If all the demand has been assigned and the convergence criteria holds then stop.
  - Otherwise:
    - Go to step 1 if all the demand has not been assigned yet
    - Or
    - Go to step 0 and start a new major iteration

![Conceptual diagram of the heuristic dynamic assignment](image-url)

Figure 1. Conceptual diagram of the heuristic dynamic assignment

Depending on how the link cost functions are defined, and whether the procedure is applied as one pass method completed when all the demand has been loaded, or it is applied as part of an iterative scheme repeated until certain convergence criterion is satisfied, it corresponds either to a “preventive” or en route dynamic traffic assignment, or to a “reactive” or heuristic equilibrium assignment. In the first case route choice decisions are made for drivers entering the network at a time interval based on...
the experienced travel times, i.e. the travel times of the previous time interval, and the link cost function is defined in terms of the average link travel times in the previous interval. Alternatively a heuristic approach to equilibrium can be based on repeating the simulation scheme a number of times and defining a link cost function including predictive terms, as proposed by Friesz et al. 1993, Xu et al. 1999. This could be interpreted in terms of a day-to-day learning mechanism.

2.1 Shortest path computation

During the simulation, the computation of shortest paths is activated at each time interval $\Delta t$. The shortest path routine used is a variation of Dijkstra's label-setting algorithm (Dijkstra 1959) that provides as a result the shortest path tree for each destination centroid. This structure of the shortest path tree provides the shortest path from every section to a destination. The penalties associated with turning movements are taken into account. Therefore, the cost labels are attached to links instead of nodes, as is usual. The link candidate list is stored as a heap data structure. During each iteration of the algorithm, the link with the minimum value of cost is removed from the heap and all links connected backward are added to the heap in the correct position.

The shortest path routine is based on link cost functions, which are evaluated/updated with the available data at the beginning of each time interval. Three type of costs functions are considered (see AIMSUN and TEDI, 2003):

- **The Initial Cost Function**, used at the beginning of the simulation when there is not yet simulated data gathered to calculate the travel times. In this case, the cost of each link is calculated as a function of the travel time in free flow conditions and the capacity of the link. There are two types of default initial cost functions. The first does not consider the vehicle types $\text{IniCost}_i$, that is, all vehicle types have the same initial cost. The second considers the vehicle types $\text{IniCost}_{i, vt}$, which means that there is an initial cost function per vehicle type. Whether to choose the initial cost function distinguishing per vehicle type or not is determined by the presence of reserved lanes in the network. By default, AIMSUN takes the initial cost function not considering vehicle types. However, if there are reserved lanes, it takes the initial cost function considering vehicle types.

- **The Dynamic Cost Function**, used when there is simulated travel time data available, that is when the simulation has already started and statistical data has been gathered. The default current cost for each section is the mean travel time, in seconds, for all simulated vehicles that have crossed the link during the last time data gathering period. As for the Initial Cost Function, there are also two types of default dynamic cost function. The first does not consider vehicle types $\text{DynCost}_i$, which means all vehicle types have the same cost. The second considers all vehicle types $\text{DynCost}_{i, vt}$, which means that there is a dynamic cost function per vehicle type. Whether to use the dynamic cost function distinguishing per vehicle type or not is determined by the presence of reserved lanes in the network.

- **User Defined Cost Functions**. The default cost functions, described above, are basically defined in terms of link travel times and don’t consider directly other wide variety of link costs, for example toll pricing, historical travel times representing driver’s experience from previous days, combinations of various link numerical attributes as for instance travel times, delay times, length and capacity, etc. Therefore, for each particular link, the user may choose between using Initial Cost Function or Dynamic Cost Function as the default cost function, or using any other cost function defined by the user using the function editor in AIMSUN when he wants to take into account other arguments represented by other link numerical attributes. The function editor allows the user to define any type of numerical function in terms of the most common mathematical functions and operators (+, -, *, /, ln, log, exp, etc.), using as function arguments any numerical attribute of the objects in the simulation model, either with fixed values (i.e. link lengths, number of lanes, speed limit on the link, etc.) or values that change during the simulation (i.e. link average speeds, average travel times, average flows, etc).

The heuristic dynamic assignment can then be defined as follows:

**Step 0**: Calculate initial shortest path(s) for each O/D pair using the defined initial costs

**Step 0.1**: Initialization:
Evaluate Initial Cost Function per each link $j$:
for each $j \in 1...L$ : $\text{Cost}_j = \text{InitialCost}_j$

**Step 0.2:** Apply Shortest Path routine:
for each destination centroid $d$:
Calculate Shortest Path Tree $SPT_d$ using $\text{Cost}_j$, $j \in 1...L$

**Step 0.3:** Identify Shortest Path from Shortest Path Tree:
for each O-D pair $i$ (from origin centroid $o$ to destination $d$)
Add to Path(s) $SP_{con}$ to $K_i$

**Step 1:** Simulate for a predefined time interval $\Delta t$ assigning to the available path $K_i$ the fraction of the trips between each O/D pair $i$ for that time interval according to the selected route choice model.

**Step 1.1:** Assignment of path probabilities:
for each O-D pair $i$:
Calculate $P_k$ using Route Choice Model, where $k \in K_i$

**Step 1.2:** Simulate for a predefined time interval $\Delta t$, generating the fraction of the vehicles between each O/D pair $i$ for that time interval, selecting randomly the path according probabilities $P_k$, $k \in K_i$

**Step 2:** Recalculate shortest path, taking into account the experimented average link travel times.

**Step 2.1:** Update Link Cost Functions:
Evaluate Dynamic Cost Function per each link $j$:
for each $j \in 1...L$ : $\text{Cost}_j = \text{DynamicCost}_j$

**Step 2.2:** Apply Shortest Path routine:
for each destination centroid $d$:
Calculate Shortest Path Tree $SPT_d$ using $\text{Cost}_j$, $j \in 1...L$

**Step 2.3:** Identify Shortest Path from Shortest Path Tree:
for each O-D pair $i$ (from origin centroid $o$ to destination $d$)
Add to Path(s) $SP_{con}$ to $K_i$

**Step 3:** If there are guided vehicles, or variable message panels proposing a rerouting, provide the information calculated in 2 to the drivers that are dynamically allowed to reroute on trip.

**Step 4:** If all the demand has been assigned then stop. Otherwise go to step 1.

### 2.2 Route choice based on a day-to-day learning mechanism

In this case the simulation is replicated $N$ times and link costs for each time interval and every replication are stored and thus at iteration $l$ of replication $j$ the costs for the remaining $l+1$, $l+2$,..., $L$ (where $L=T/\Delta t$, being $T$ the simulation horizon and $\Delta t$ the user defined time interval to update paths and path flows) time intervals for the previous $j-1$ replications can be used in an anticipatory day-to-day learning mechanism to estimate the expected link cost at the current iteration. Let $s^{jl}_{ia}(v)$ be the current cost of link $a$ at iteration $l$ of replication $j$, then the average link costs for the future $L-l$ time intervals, based on the experienced link costs for the previous $j-1$ replications is:

$$s^{j+l}_{ia}(v) = \frac{1}{j-1} \sum_{m=1}^{j-1} s^{m,l+1}_{ia}(v), \quad i = 1,...,L-l,$$  \hspace{1cm} (4)
The “forecasted” link cost can then be computed as:

\[ \tilde{c}_d^{j,i,l+1}(v) = \sum_{i=0}^{L-1} \alpha_i \tilde{c}_d^{j,i,l+1}(v) \text{ where } \sum_{i=0}^{L-1} \alpha_i = 1, \alpha_i \geq 0, \forall i; \text{ are weighting factors} \] (5)

The resulting cost of path \( k \) for the \( i \)-th OD pair is

\[ \tilde{S}_k^{l+1}(v) = \sum_{a \in A} \tilde{c}_d^{j,i,l+1}(v) \delta_{ak} \] (6)

where, as usually \( \delta_{ak} \) is 1 if link \( a \) belongs to path \( k \) and 0 otherwise. The path costs \( \tilde{S}_k^{l+1}(v) \) are the arguments of the route choice function (logit, C-logit, proportional, user defined, etc.) used at iteration \( l+1 \) to split the demand \( g_i^{l+1} \) among the available paths for OD pair \( i \).

In the computational experiments discussed in this paper a simplified version consisting of a link cost function defined as:

\[ c_{it}^{k+1} = \lambda c_{it}^k + (1 - \lambda) \tilde{c}_{it}^k \] (7)

Where \( c_{it}^{k+1} \) is the cost of using link \( i \) at time \( t \) at iteration \( k+1 \), and \( c_{it}^k \) and \( \tilde{c}_{it}^k \) correspond respectively to the expected and experienced link costs at this time interval from previous iterations.

### 2.3 Route Choice Models

In the proposed network loading mechanism based on microscopic simulation vehicles follow paths from their origins in the network to their destinations. So the first step in the simulation process is to assign a path to each vehicle when it enters the network, from its origin to its destination. This assignment, made by a path selection process based on a discrete route choice model, will determine the path flow rates.

Given a finite set of alternative paths, the path selection calculates the probability of each available path and then the driver’s decision is modeled by randomly selecting an alternative path according to the probabilities assigned to each alternative. Route choice functions represent implicitly a model of user behavior, that emulates the most likely criteria employed by drivers to decide between alternative routes in terms of the user’s perceived utility (or, properly speaking, a disutility, or cost in the case of trip decisions) defined in terms of perceived travel times, route lengths, expected traffic conditions along the route, etc.

The simulation experiments reported in this paper have been implemented in AIMSUN selecting the Logit, C-Logit and Proportional route choice functions from the default route choice functions available in the simulator. The Multinomial Logit route choice model defines the choice probability \( P_k \) of alternative path \( k \), \( k \in K_i \), as a function of the difference between the measured utilities of that path and all other alternative paths:

\[ P_k = \frac{e^{\theta V_i}}{\sum_{l \in K_i} e^{\theta V_l}} = \frac{1}{1 + \sum_{l \neq k} e^{\theta (V_i - V_l)}} \] (8)

where \( V_i \) is the perceived utility for alternative path \( i \) (i.e. the opposite of the path cost, or path travel time), and \( \theta \) is a scale factor that plays a two-fold role, making the decision based on differences between utilities independent of measurement units, and influencing the standard error of the distribution of expected utilities, determining in that way a trend towards utilizing many alternative routes or concentrate in very few routes, becoming in that way the critical parameter to calibrate how the logit route choice model leads to a meaningful selection of routes or not.

A drawback reported in using the Logit function is the observed tendency towards route oscillations in the routes used, with the corresponding instability creating a kind of flip-flop process. According to our experience there are two main reasons for this behavior. The properties of the Logit function, which
and the inability of the Logit function to distinguish between two alternative routes when there is a high degree of overlapping.

The instability of the routes used can be substantially improved when the network topology allows for alternative routes with little or no overlapping at all, playing with the shape factor of the Logit function and re-computing the routes very frequently. However, in large networks where many alternative routes between origin and destinations exist, and some of them exhibit a certain degree of overlapping the use of the Logit function may still exhibit some weaknesses. To avoid this drawback the C-Logit model, (Cascetta et al., 1996; Ben-Akiva and Bierlaire, 1999), has been implemented.

In this model, the choice probability $P_k$ of each alternative path $k$ belonging to the set $K_i$ of available paths connecting the i-th OD pair, is defined by:

$$P_k = \frac{e^{\theta(V_i - CF_k)}}{\sum_{l \in K_i} e^{\theta(V_i - CF_l)}}$$  \hspace{1cm} (9)

where $V_i$ is the perceived utility for alternative path $i$, i.e. the opposite of the path cost, and $\theta$ is the scale factor, as in the case of the Logit model. The term $CF_k$, denoted as ‘commonality factor’ of path $k$, is directly proportional to the degree of overlapping of path $k$ with other alternative paths. Thus, highly overlapped paths have a larger CF factor and therefore smaller utility with respect to similar paths. $CF_k$ is calculated as follows:

$$CF_k = \beta \cdot \ln \left( \sum_{l \in K_i} \left( \frac{L_{lk}}{L_l^{\frac{1}{2}} L_k^{\frac{1}{2}}} \right)^\gamma \right)$$  \hspace{1cm} (10)

where $L_{ik}$ is the length of arcs common to paths $i$ and $k$, while $L_i$ and $L_k$ are the length of paths $l$ and $k$ respectively. Depending on the two factor parameters $\beta$ and $\gamma$, a greater or lesser weighting is given to the ‘commonality factor’. Larger values of $\beta$ means that the overlapping factor has greater importance with respect to the utility $V_i$; $\gamma$ is a positive parameter, whose influence is smaller than $\beta$ and which has the opposite effect. The utility $V_i$ used in this model for path $i$ is the opposite of the path travel time $t_t$, (or path cost depending on how has been defined by the user).

Other option is the estimation of the choice probability $P_k$ of path $k$, $k \in K_i$, in terms of a generalization of Kirchoff’s laws given by the function

$$P_k = \frac{CP_k^{-\alpha}}{\sum_{l \in K_i} CP_l^{-\alpha}}$$  \hspace{1cm} (11)

where $CP_l$ is the cost of path $l$, $\alpha$ is in this case the parameter whose value has to be calibrated.

### 3. Initial K-Shortest Paths

At the beginning of the simulation, using the Initial Cost function, one shortest path tree is calculated per each destination centroid, so during the first interval all vehicles are assigned to the same alternative. In order to start considering more than one alternative, as a way to anticipate the assignment process, at the beginning of the simulation, k-shortest path tree are calculated. The problem of enumerating, in order of increasing length, the k shortest paths has received considerable attention in the literature, Bellman 1960, Dreyfus 1969, Fox 1973, Eppstein 1994,1999. The different algorithms to solve this problem Eppstein 1994, Eppstein 1999, Jimenez et al. 1999, Martins 1984, Martins et al. 1996, are based in, after computing the shortest path from every node in the graph, the algorithm builds a graph representing all possible deviations from the shortest path. Therefore, all k shortest path obtained as a result, use the same length arc, associated to a cost function, and in our case the calculation of the k shortest path has to anticipate the evolution of the arc costs considering the traffic flow assigned to each path.
The algorithm calculates every iteration a new shortest path until the number of shortest path available reaches the parameter \( \text{MaxKSP} \). The figure 2 depicts the generic scheme of the algorithm that iteratively:

- evaluates the cost function in each arc (first iteration the cost function is the travel time in free-flow conditions)
- calculates a new shortest path.
- determine the path flow (using an incremental loading procedure, described below) and update the flow in each arc

The components of this algorithm are:

- **Shortest Path Algorithm**
  The computation of the shortest path corresponds a variation of Dijkstra’s label setting algorithm.

- **Arc Cost Function**
  The cost of using arc \( a \) \( (sa(va)) \) is a function of the flow in arc \( a \) \( v_a \):
  \[
  s_a(v_a) = n_0 \left( 1 + \beta \frac{v_a - \alpha}{c_a} \right)
  \]

- **Incremental Loading Algorithm**
  The path flow rates in the feasible region \( \Omega \) satisfy the conservation flow and non-negativity constraint (where the traffic demand of O-D pair \( i \) is denoted by \( g_i \)). That is:
  \[
  \Omega = h^i_k : \sum_{k \in K_i} h^i_k = g_i, i \in I; h^i_k \geq 0
  \]

For each O-D pair \( i \in I \) and path \( h^i_k \), evaluate the path flow assigned to path \( h^i_k \) \( k \in K_i \) at iteration \( n \) \( h^i_k(n) \):

- \( h^i_k(n) = h^i_k(n-1) + \lambda(g_i - h^i_k(n-1)) \)
- \( h^i_k(n) = h^i_k(n-1) - \lambda(h^i_l(n-1)), l = 1 \ldots (k-1) \)

where \( \lambda = \frac{1}{n+1} \)

---

**Figure 2: Generic scheme of K-Shortest Path Algorithm**

The algorithm to calculate the \( k \)-shortest path can be stated as follows (\( n \) is the iteration index and \( k \) is the shortest path index):

**Step 0:** Initialization : \( n=0 \) and \( k=1 \)

Compute the \( k \)-th shortest path based on the free-flow travel times.
For each O-D pair \( i \in I \), assign \( h^i_k(n) = g_i \)

**Step 1:** Compute the flow \( v_a \) for each arc \( a \):

\[
v_a = \sum_{i \in I} \sum_{k \in K_i} \delta_{a,k} h^i_k
\]

where \( \delta_{a,k} = \begin{cases} 
1, & \text{arc } a \text{ belongs to path } k \text{ of O-D pair } i \\
0, & \text{otherwise}
\end{cases} \)

Evaluate the cost function of each arc \( a \) \( (s_a(v_a)) \).

**Step 2:** \( k = k + 1, n = n + 1 \)

Compute the k-th shortest path based cost function of each arc \( a \) \( (s_a(v_a)) \).

Incremental Loading: For each O-D pair \( i \in I \), evaluate:

\[
h^i_k(n) = h^i_k(n - 1) + \lambda(g_i - h^i_k(n - 1))
\]

for \( l = 1 \) to \( (k - 1) \)

\[
h^i_l(n) = h^i_l(n - 1) - \lambda(h^i_l(n - 1)), \quad l = 1 \ldots (k - 1)
\]

where \( \lambda = \frac{1}{n + 1} \)

**Step 3:** If \( k \) is equal to total number of shortest path \( MaxKSP \) then STOP
Otherwise, return Step 1

The figure 3 depicts an example of 3 initial k –shortest paths, in the model of the Preston network.

4. COMPUTATIONAL RESULTS

No formal convergence proof can be given for the proposed heuristic dynamic assignment algorithm, since the heuristic network loading process based on microscopic simulation does not have an analytical form. A set of simulation experiments has been designed and conducted to explore empirically whether the described assignment process, depending on how it is implemented, can be associated to a heuristic realization of a preventive or a reactive dynamic assignment, assuming that a proper selection of a route choice model with the right values for the \( \theta, \beta, \gamma \) or \( \alpha \) parameters, depending on the model, should lead to the realization of some equilibrium. A way of measuring the progress towards the equilibrium in an assignment, and therefore qualify the solution, is the relative gap function, \( R_{gap}(t) \), Florian et al. 2001, Janson, 1991, that estimates at time \( t \) the relative difference between the total travel time actually experienced and the total travel time that would have been experienced if all vehicles had the travel time equal to the current shortest path:

\[
R_{gap}(t) = \frac{\sum_{i \in I} \sum_{k \in K_i} h_k(t)[s_k(t) - u_i(t)]}{\sum_{i \in I} g_i(t)u_i(t)}
\]

(12)

Where \( u_i(t) \) are the travel times on the shortest paths for the i-th OD pair at time interval \( t \), \( s_k(t) \) is the travel time on path \( k \) connecting the i-th OD pair at time interval \( t \), \( h_k(t) \) is the flow on path \( k \) at time \( t \), \( g_i(t) \) is the demand for the i-th OD pair at time interval \( t \), \( K_i \) is the set of paths for the i-th OD pair, and \( I \) is the set of all OD pairs.
The figures 4, 5, 6 and 7, and 8 and 9 respectively, depict the test networks, and time evolution of the $R_{gap}(t)$ function for various Route Choice functions, for the preventive, or en-route version of the assignment procedure, using a K-shortest path algorithm, for the test models of:

- The borough of Amara in the City of san Sebastián in Spain. A model with 365 road sections, 100 nodes and 225 OD pairs.
- The model of Brunnsvik network in Stockholm. This model has 493 road sections, 260 nodes and 576 OD pairs.
- The model of the Preston City Centre in UK. The model has 1375 road sections, 188 nodes and 1156 OD pairs.

Figure 3. 3 initial k-shortest paths in Preston City centre Model

In these figures Logit $n$, corresponds to the above defined Logit function with value $n$ for the shape parameter $\theta$, proportional corresponds to a path probability inversely proportional to the path cost. The expected role of the $\theta$ parameter in terms of the $R_{gap}$ function becomes evident in the combination of the logit function with the assignment procedure. Improper choices of the parameter values tend to produce a bang-bang effect consequence of the tendency to move most of the flow to the current shortest path, as the oscillations of the $R_{gap}$ function show, while a more appropriate $\theta$ value ($\theta = 30$ in Amara, or $\theta = 900$ in Brunnsvik) not only smooths out significantly the $R_{gap}$ oscillations but also shows that a path selection with acceptable path costs differences (a 10% in Amara and around a 1% in Brunnsvik).

The figures 10 and 11, depict the time evolution of the $R_{gap}$ function for the same logit route choice function, for the reactive version of the assignment procedure using the costs as defined in (1), at iteration $k=20$, and $\lambda=0.25$, 0.5 and 0.75 respectively, for $\theta$ values of 30 in Amara, and 900 in Brunnsvik. $R_{gap}$ values tend almost to zero, as expected in equilibrium terms, and the variations for the various values of $\lambda$ show that $\lambda=0.75$ is the best.

A side result of the method explored in this paper is the guidance that the computational results provide assisting in the calibration of the $\theta$ or $\alpha$ parameters, depending on the route choice function selected, based on the assumption that, as far as the described assignment process, depending on how it is implemented, can be associated to a heuristic realization of a preventive or a reactive
dynamic assignment, a proper route selection should lead to a realization of some equilibrium, and the progress towards such equilibrium is measured in terms of the Rgap function.

Figure 4. AIMSUN model of Amara

Figure 5. Time evolution of the Rgap function for various Route Choice functions for Amara model (Preventive case)
Figure 6. AIMSUN model of Brunnsviken network in Stockholm

Figure 7. Rgap function for Brunnsviken (Preventive case)
Figure 8. AIMSUN model of Preston City Centre

Figure 9. Rgap for Preston (Preventive case)
Figure 10. Rgap for Amara (Reactive case)

Figure 11. Rgap for Brunnsviken (Reactive case)
5. CONCLUSIONS

Assuming that the “dynamic equilibrium” exists the empirical results show that a proper time varying k-shortest paths calculation, with a suitable definition of link costs, and adequate stochastic route choice functions, using a microscopic network loading mechanism, achieves a network state that can replicate acceptably the observed flows over the simulation horizon, and led to a reasonable set of used paths between OD pairs as the oscillations within a narrow band of the empirical Rgap function indicates.

6. REFERENCES


