

Modelling Advanced Transport Telematic Applications with Microscopic Simulators: the Case of AIMSUN2

J. Barceló¹, J. Casas², J.L. Ferrer¹ and D. García²

¹ Laboratori de Simulació i Investigació Operativa, Departament d'Estadística i Investigació

Operativa, Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028 Barcelona, Spain

e-mail: lios@eio.upc.es

² TSS-Transport Simulation Systems, Tarragona 110-114, 08015 Barcelona, Spain

e-mail: info@tss-bcn.com

The simulation of Advanced Transport Telematic Applications requires specific modelling features, which have not been usually taken into account in the design of microscopic traffic simulation models. This paper discusses the general requirements of some of these applications, and describes how have they been implemented in the microscopic traffic simulator AIMSUN2.

1. Introduction

Microscopic traffic simulators are simulation tools that emulate realistically the flow of vehicles on a road network. The main modelling components of a microscopic traffic simulation model are: an accurate representation of the road network geometry, a detailed modelling of individual vehicles behaviour, and an explicit reproduction of traffic control plans. The primary attention has been paid usually to the proper modelling and calibration of all these model components, namely the car-following, gap acceptance, lane change, and other internal models which along with other modelling parameters accounting for attributes of the physical system entities, allow the microscopic simulation model reproduce flow, speed, occupancies, travel time, average queue lengths, etc. with enough accuracy to consider the model valid.

The advent of the Advanced Transport Telematic Applications made possible by combining the developments in informatics and telecommunications applied to transportation problems, has created new objectives and requirements for micro-simulation models. Quoting from Deliverable D3 of the SMARTTEST Project [1]: "The objective of micro-simulation models is essentially, from the model designers point of view, to quantify the benefits of Intelligent Transportation Systems (ITS), primarily Advanced Traveller Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS). Micro-simulation is used for evaluation prior to or in parallel with on-street operation. This covers many objectives such as the study of dynamic traffic control, incident management schemes, real-time route guidance strategies, adaptive intersection signal controls, ramp and mainline metering, etc. Furthermore some models try to assess the impact and sensitivity of alternative design parameters".

The current trend in the development of Advanced Transport Telematic Applications, either real-time adaptive, or based on other specific approaches, is far from being standardised. It is therefore an exercise of dubious utility to try to integrate them in a fixed way in a microscopic traffic simulator. The relative gain achieved by including any of these, as an in-built function of the microsimulator is limited to simulating, on an easier way, those road networks on which the selected application is operating. However there would be no means of simulating other systems with that microsimulator. This is true whenever we address the problem of simulating adaptive traffic control systems as, for example, SCOOT, SCATS, vehicle actuated, control systems giving priority to public transport, etc., Advanced Traffic Management Systems (using VMS, traffic calming strategies, ramp metering policies, etc), Vehicle Guidance Systems, Public Transport Vehicle Scheduling and Control Systems or applications aimed at estimating the environmental impacts of pollutant emissions, and energy consumes. The main question then is: How can these Advanced Transport Telematic Applications be properly evaluated and tested by simulation?

From a conceptual point of view the operation of these modern systems can be described as follows: for certain applications the road network is suitably equipped with traffic detectors of various technologies (loop detectors, image processing detectors, etc.), with a specific layout depending on the requirements of the control approach. They supply the necessary real-time traffic data (flows, speeds, occupancies, etc) with the required degree of aggregation. These real-time traffic measurements feed the logic of the traffic control or management system which, after suitable processing, makes ad hoc control decisions: e.g. extend the green phase, change to the red phase, apply some traffic calming strategies, etc.. Other applications as, for example vehicle guidance, public transport monitoring systems, or the evaluation of environmental impacts, require the access to vehicle data (position, speed, acceleration, etc.), to emulate the up-link messages in vehicle guidance applications, the vehicle

tracking for the public transport monitoring or fleet management systems, or simply to provide the required data for certain fuel consumption or pollutant emissions models. To evaluate and test any of these systems a microsimulator must be capable of incorporating in the model the corresponding traffic devices as objects: i.e. detectors, traffic lights, VMS, etc. It must also emulate their functions: provide the specific traffic measurements at the required time intervals, increase the phase timing in a given amount of time, implement a traffic calming strategy (slow down the speed on a road section, recommend an alternative route, etc). How can such evaluations be done by simulation without explicit in-built modelling of the specific Advanced Telematic Application?

2. The GETRAM/AIMSUN2 Microsimulator

2.1 GETRAM

GETRAM (Generic Environment for Traffic Analysis and Modelling), [2-3], is a simulation environment comprising a traffic network graphical editor (TEDI), a microscopic traffic simulator (AIMSUN2), a network data base, a module for storing results and an Application Programming Interface to aid interfacing to assignment models and other simulation models, as for example the macroscopic traffic assignment EMME/2 system, [4]. The functional structure of the systems is depicted in figure 1.1.

2.2 AIMSUN2 microscopic simulator

AIMSUN2 (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks), [5], is a microscopic traffic simulator that can deal with different traffic networks: urban networks, freeways, highways, ring roads, arterial and any combination of them. AIMSUN2 simulates traffic flows either based on input traffic flows and turning proportions, or on O-D matrices and route selection models. In the former, vehicles are distributed stochastically around the network, whereas in the latter vehicles are assigned to specific routes from the start of their journey to their destination. Different types of traffic control can be modelled in AIMSUN2: traffic signals, junctions without traffic signals (give way or stop signs) and ramp metering. Vehicle behaviour models (car following, lane change, gap acceptance, etc.) are function of several parameters that allow modelling of different types of vehicles: cars, buses, trucks, etc. They can be classified into groups, and reserved lanes for given groups can also be taken into account

AIMSUN2 can also simulate any kind of measurable traffic detector: counts, occupancy and speed. AIMSUN2 has a user-friendly interface through which the user can define the simulation experiment. It also provides a picture of the network and an animated representation of the vehicles in it. The user has an overview of what is happening in the network that aids performance analysis. Through the interface, the user may access any information in the model and define traffic incidents before or during the simulation run.

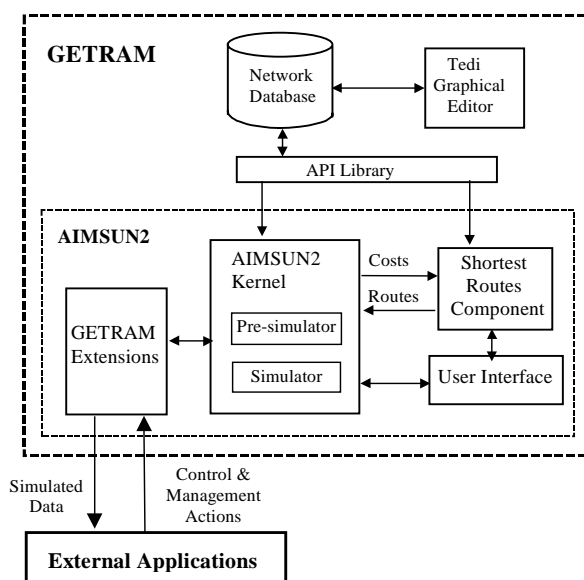


Figure 1.1

In the variable mode, regardless of the Route Choice model used, there are two types of driver’s behaviour with respect to the route assignment: Static and Dynamic, which refers to whether or not a vehicle can modify the actual path en-route as new paths become available during the trip.

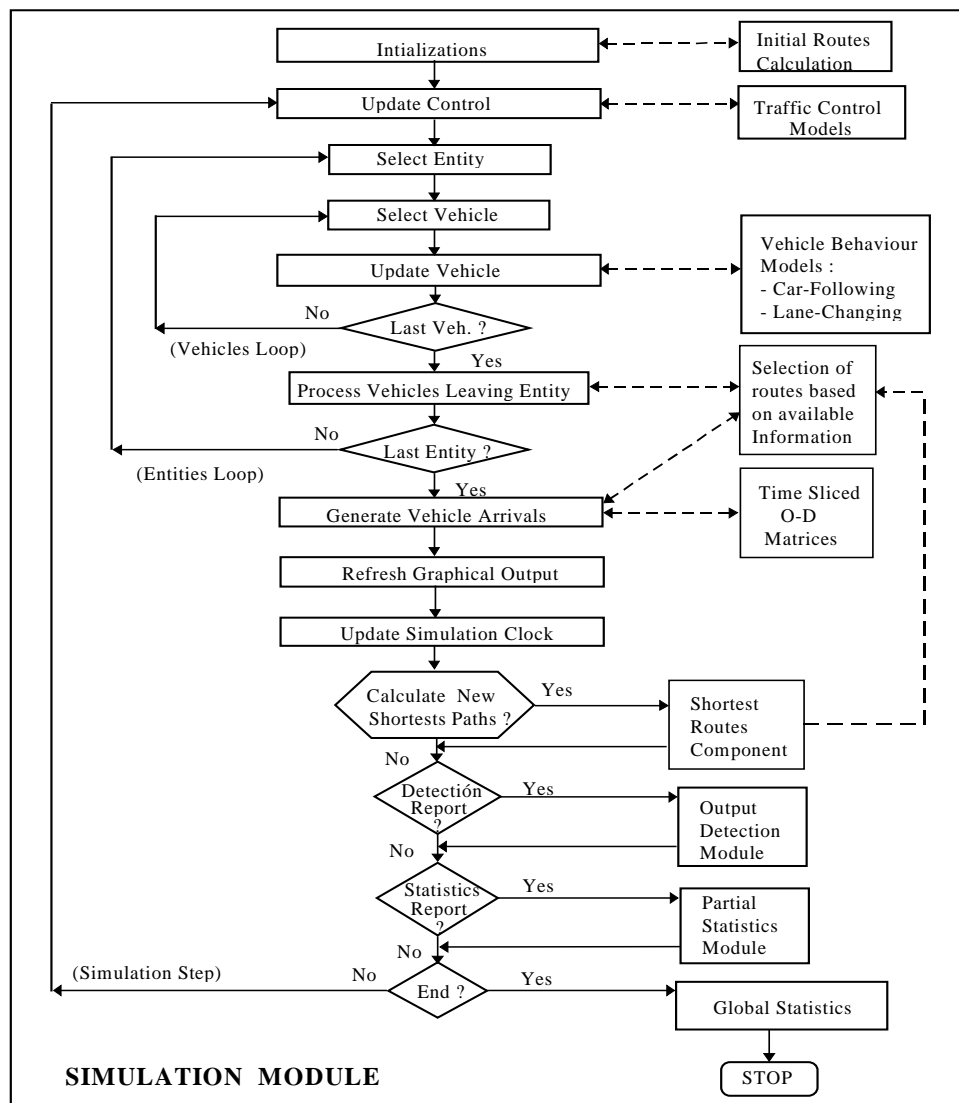


Figure 2: Logic of the Simulation Process in AIMSUN2

2.3.1 Shortest Routes Component

This route based version of AIMSUN2 [3] becomes a simulation platform for networks containing an ATMS/ATIS component. These centres provide information that changes during the course of time and the vehicles react by possibly using different alternative routes. This implies that the simulator needs to store the current shortest routes from the beginning of every section to all destinations (whether these are sections or centroids). It is also needed to store the routes that different vehicles wish to follow. Since a vehicle may miss its turn, the storage of a route with a vehicle is not a practical solution. Hence, one needs to keep all previously generated routes as long as there are vehicles using them. For each destination and instant in time, the routes are stored as a tree that allows knowing how to reach the destination from any section of the network. We also attach to this tree a field that counts the number of vehicles using it. When this counter is empty, the tree may be deleted.

The shortest routes component takes into account the turning penalties as the different turning movements at the end of a section have in general unequal travel times (e.g. left turn, going straight, etc). The procedure that we use to compute the shortest routes to a destination (either a centroid node or a section) uses a network where an arc,

connecting two nodes, models a section. A special arc connecting the beginning of the turning to its end models a turning movement. The computation of shortest routes uses a label setting method, where the labels are associated with an arc. The network is constructed only once before the start of the simulation.

During the simulation, the computation of shortest routes is launched at certain time steps. This is determined usually in a periodic manner, with a period that depends on the length of the section and on the level of congestion. Very frequent calculations are unnecessary, while a low frequency of calculations leads to a state where drivers have a very low level of information. Hence, one seeks an intermediate value for this periodicity. AIMSUN2 is so equipped that this parameter may be changed and experiments with different values may be studied and evaluated.

The shortest route routine is a variation of Dijkstra's label setting algorithm. It provides the shortest routes from the start of every section to all destinations. The penalties associated with turning movements are taken into account. Hence, the cost labels are attached to sections instead of nodes, as it is usual. The arc candidate list is stored as a heap data structure. During each iteration of the algorithm, the section with minimum value is removed from the heap and the heap is restored by using efficient operations. As a new section is reached, one adds it to the heap in the right position.

Section Cost Functions

Two types of section cost functions are used for calculating the shortest path trees, depending on whether or not there are simulated data available to be used for. These are the Initial Cost Function and the Current Cost Function. In both cases, the cost function represents section travel time in seconds, including the penalty of the turning movement, if it exists.

The *Initial Cost Function* is applied at the beginning of simulation when there is no simulated data gathered to calculate the travel times. In this case, the cost of each section is calculated as a function of the travel time in free flow conditions and the capacity of the section. The travel time in free flow conditions is the time it would take a vehicle, travelling at the maximum allowed speed of the section, to cross the section.

The initial cost of each section, $IniCost(s)$, is calculated as follows:

$$IniCost(s) = TravelTFF(s) + TravelTFF(s) \times \varphi \times \left(1 - \frac{Capacity(s)}{MaxCapacity} \right)$$

where:

$TravelTFF(s)$ is the travel time, in seconds, of section s in free flow conditions. It is calculated as $Length(s)/SpeedLimit(s)$.

$Capacity(s)$ is the capacity of section s , in vehicles per hour.

$MaxCapacity$ is the maximum capacity of any section in the network.

φ : Capacity weight. It is a user-defined parameter that allows the user to control the influence that the section capacity has in the cost in relation with the travel time.

The *Current Cost Function* can only be applied when there is some simulated travel time data available, and therefore it can not be used at the beginning of the simulation but only when the simulation has already started and some statistical data has been gathered.

The current cost for each section, $CurrCost(s)$, is the mean travel time, in seconds, for all simulated vehicles that have crossed the section during the last statistical gathering period ($TravelTime(s)$). As there may be situations in which any vehicle has not crossed a section, the following algorithm is applied to calculate $CurrCost(s)$:

```

if (Flow(s) > 0) then
    CurrCost(s) = TravelTime(s)
else
    if (there is any vehicle stopped) then
        CurrCost(s) = Maximum (AvgTimeIn, IniCost(s))
    else
        CurrCost(s) = IniCost(s)
    endif
endif

```

According to this algorithm, when some vehicle has crossed the section during the last statistical period ($Flow(s) > 0$), the cost is the simulated mean travel time. In case that no vehicle has crossed the section we distinguish the case of a totally congested section from the case of an empty section. In the first case, the cost is calculated as the maximum between the Initial Cost and the average waiting time for the vehicles in front of the queue in the section ($AvgTimeIn$). In the second case, the cost is taken as the initial cost.

2.3.2 Fixed Routes Mode

In the Fixed Routes Mode, shortest path trees are calculated from every section to every destination centroid at the beginning of the simulation. Then, during the simulation, vehicles are generated at origin centroids and assigned to the shortest route to their destination centroid. There is no need for Route Choice Model as there are not alternative routes. No new routes are recomputed during simulation; therefore all vehicles always follow the shortest path and no decisions about changing to another path can be made during the trip.

Depending on the type of cost function used for the initial shortest path calculations, there are two alternatives Fixed Routes Models. They are called the *Fixed-Distance* and the *Fixed-Time* Models.

In the *Fixed-Distance Model* the paths are calculated at the beginning of simulation taking as cost of each section the Initial Cost, either if there is a warm-up period defined or not. In case that there is warm-up period, no new shortest paths are calculated when it ends, and therefore, during the stationary simulation period the same shortest path trees are used. Figure 3 illustrates when the shortest paths (SP) are calculated in a time diagram of the simulation period, both in case that there is warm-up defined or not.

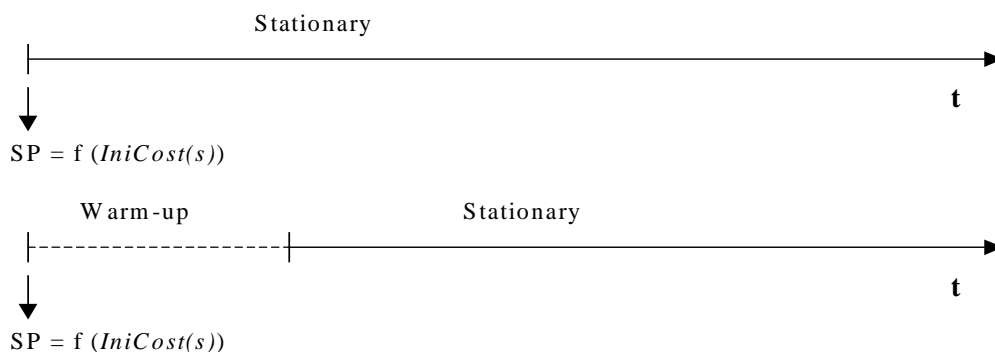


Figure 3: Calculation of Shortest Paths in a Fixed Distance Model

The *Fixed-Time Model* works similar to the *Fixed-Distance Model* except in the case that there is a Warm-up period defined. In this case, initial paths are calculated at the beginning of the Warm-up in the same way using the Initial Costs. But now, when the Warm-up period is over and the stationary simulation starts, new initial paths are calculated taking as section costs the Current Cost Function, which is calculated using the statistical data gathered during the simulation warm-up. Figure 4 illustrates the shortest paths (SP) calculations in a time diagram of the simulation period, both in case that there is warm-up defined or not..

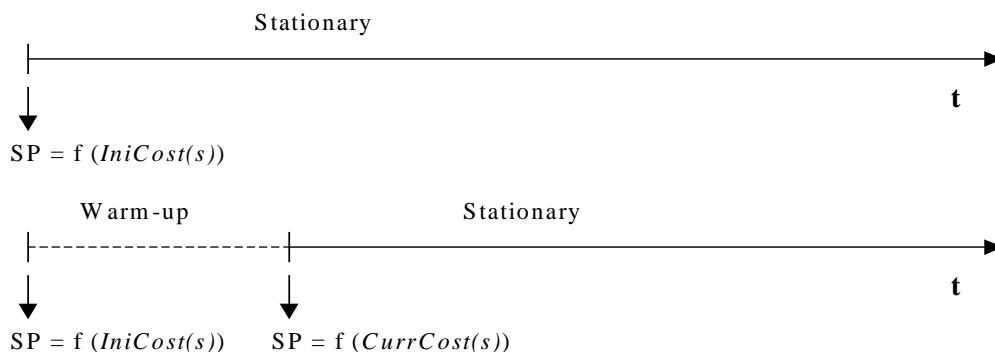


Figure 4: Calculation of Shortest Paths in a Fixed Time Model

In the Fixed-Distance Model the cost is very theoretical and does not take into account the network congestions, only the length of the paths and the allowed speed. We call it Fixed-*Distance* to denote that the cost is mainly based on the distances, together with the speed limits and the capacity, but not on the traffic conditions at any time. In the Fixed-Time Model the cost is influenced by the traffic conditions at certain time and therefore it represents more accurately the travel time. To denote this difference, we call it Fixed-*Time*.

2.3.3 Variable Routes Mode

In the Variable Routes Mode the simulation process includes an initial computation of shortest routes going from every section to every destination, a shortest route component which calculates periodically the new shortest routes according to the new travel times provided by the simulator, and a route selection model.

The simulation procedure can be characterised as follows:

1. Calculate initial shortest routes, taking as costs the estimated travel times for each section (i.e. length of section / speed limit).
2. Simulate for a period (e.g. 5 minutes) using available routes information and obtain new average travel times as a result of the simulation.
3. Recalculate shortest routes, taking into account the new travel times.
4. Add the new information calculated in 3 to the knowledge of the drivers.
5. Go to step 2.

At the beginning of the simulation, shortest path trees are calculated from every section to each destination centroid, taking as section costs the Initial Cost Function, as in the previous case. If there is Warm-up period defined, these paths are calculated at the beginning of the Warm-up, if not, they are calculated at the beginning of the stationary simulation period.

During simulation, new routes are recomputed every time interval taking as section costs the simulated travel times obtained for each section during the last interval, this is the Current Cost Function explained before. Figure 5 illustrates when are the shortest paths (SP) calculated along the simulation period and what are the cost functions used, both in case that there is warm-up defined or not..

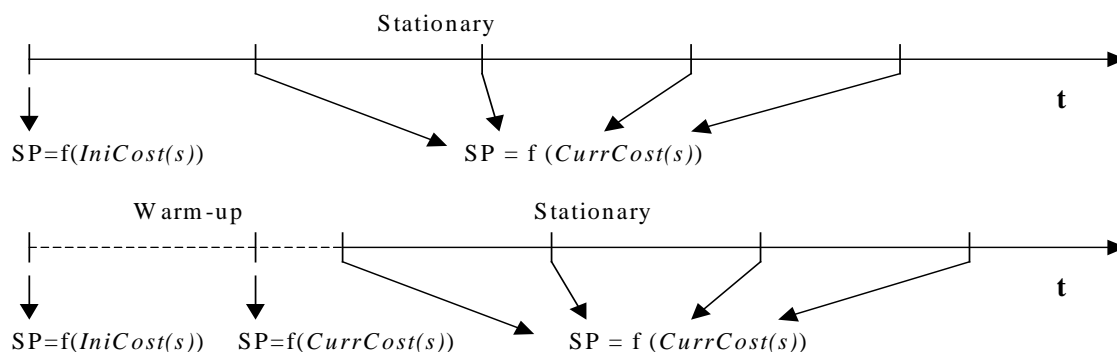


Figure 5: Calculation of Shortest Paths in a Variable Routes Mode

The user may define the time interval for recalculation of paths and the maximum number of path trees that wishes to maintain during the simulation. When the maximum number of path trees (K) is reached, the oldest paths will be removed as soon as no vehicle keep following them. It is assumed that vehicles only choose among the most recent K path trees, therefore, the oldest ones will become obsolete and disused.

Static versus Dynamic Route assignment Models

Vehicles are initially assigned to a route from a set of available routes on a probabilistic way. Apart from the initial assignment of route, which is made at the vehicle's depart time, there is the possibility of making a route reassignment during the trip. This is called the Dynamic route choice model, as opposite to the Static one.

In the Dynamic route choice model a guided vehicle can make a new decision about what route to follow at any time along their trip, whenever there are new shortest routes available. In the Static model, a vehicle will always follow its initially selected route until reaching the destination, although new shortest route could be available during the trip. Note that in the Dynamic model only guided vehicles can take the decision of changing to a new shortest route during the trip, as it is supposed that information is only available for those equipped vehicles. Regarding this, there is a parameter for each vehicle type that gives the percentage of guided vehicles.

The behaviour of the driver in response to information acquisition may be modelled in different ways: as a shortest route follower, a boundedly rational user or route choice based on discrete path choice models. This opens the way to a wide variety of applications in the context of ATMS/ATIS systems.

2.3.4 Route Choice Models

Currently there are two Choice Models implemented, which are used either when assigning the initial path for a vehicle at the beginning of its trip or when having to decide whether or not to change path en-route in the dynamic modelling. These models are the Binomial and the Multinomial Logit models.

Binomial Model

A Binomial ($k-1, p$) distribution is taken to find the probability of selecting each path. Parameter k is the number of available paths and p is the “success” probability. This model does not consider the travel costs in the decision process, but only the time at which the path was calculated. Selecting a small p will make that oldest paths will be more likely used while selecting high values of p , the most recent paths will be more frequently taken.

For instance, if we want to keep three alternative paths and let the newest paths be more used, we may define $k=3$ and $p=0.9$. Then the feasible values for $X = \text{Binomial}(2, 0.9)$ are $X = 0$, $X = 1$ and $X = 2$, which are respectively associated to the 3 last calculated paths. Then, the probability of selecting the oldest path is $P(X=0) = 0.01$, the probability of selecting the second path is $P(X=1) = 0.18$ and the probability of selecting the newest path is $P(X=2) = 0.81$.

Multinomial Logit Model

The choice probability P_k of a given alternative path k can be expressed as a function of the difference between the measured utilities of that path and all other alternative paths:

$$P_k = \frac{1}{1 + \sum_{l \neq k} e^{(v_l - v_k) \theta}}$$

where V_i is the perceived utility for alternative path i . We have taken $V_i(t) = -Ti(t)$, which is the current simulated travel time for path i at time t , measures in hours. This time is calculated as the sum of the current costs of all the sections composing the path (CurrCost(s) function as explained above).

We assume that the utility U_k^{rs} of route k between origin r and destination s is given by:

$$U_k^{rs} = -\theta t_k^{rs} + \varepsilon_k^{rs}$$

Where:

θ is a shape or scale factor parameter

t_k^{rs} is the expected travel time on route k from r to s , and

ε_k^{rs} is a random term

The underlying modelling hypothesis is that random terms ε_k^{rs} are independent identically distributed GUMBEL variates. Under these conditions the probability of choosing route k amongst all alternative routes from r to s is given by the logistic distribution:

$$P_k^{rs} = \frac{e^{-\theta t_k^{rs}}}{\sum_l e^{-\theta t_l^{rs}}} = \frac{1}{1 + \sum_{l \neq k} e^{-\theta(t_l^{rs} - t_k^{rs})}} \quad (1)$$

The scale factor θ plays a twofold role making independent of the measurement units the decision based on differences between utilities, and influencing the standard error of the distribution of expected travel times:

$$\text{Var}(t_k^{rs}) = \frac{\pi^2}{6\theta^2}$$

that is:

$\theta < 1$ high perception of the variance, in other words a trend to utilise many alternative routes

$\theta > 1$ alternative choices are concentrated in very few routes

For example, given four alternative routes with expected travel time of $T_1=12$ minutes, $T_2=15$ minutes, $T_3=16$ minutes and $T_4=18$ minutes, the corresponding probabilities according to (1) when $\theta = 1$ are: $P_1=0.93407$, $P_2=0.04650$, $P_3=0.01710$ and $P_4=0.00231$, whereas if $\theta = 0.5$ the probabilities are: $P_1=0.71009$, $P_2=0.15844$, $P_3=0.09610$ and $P_4=0.03535$.

Parameter, or scale factor θ in AIMSUN2 is a user defined parameter that can consequently be used to adjust the effect that small changes in the travel times may have on the driver's decisions.

2.4 Modelling of vehicle movement

During their journey along the network, the vehicles are updated according to vehicle behaviour models: "Car-Following" and "Lane-Changing". Drivers tend to travel at their desired speed in each section but the environment (i.e. preceding vehicle, adjacent vehicles, traffic signals, signs, blockages, etc) conditions their behaviour.

Simulation time is split into small time intervals named simulation cycle or simulation step (Δt). Each simulation cycle, the position and speed of every vehicle in the system is update in accordance to the following algorithm:

if (it is necessary to change lanes) then
Apply Lane-Changing Model

endif

if (the vehicle has not changed lanes) then
Apply Car-Following Model

endif

Once all vehicles have been updated for the current cycle, vehicles scheduled to arrive during this cycle are introduced into the system and next vehicle arrival times are generated.

2.4.1 Car Following Model

The car following model implemented in AIMSUN2 is based on the Gipps model [7], and can be considered as an ad hoc evolution of this empirical model in which the model parameters are not global but determined by the influence of local parameters depending on the "type of driver" (limit speed acceptance of the vehicle), the geometry of the section (speed limit on the section, speed limits on turnings, etc.), the influence of vehicles on adjacent lanes, etc. It consists on two components: acceleration and deceleration. The first represents the intention of a vehicle to achieve certain desired speed, while the second reproduces the limitations imposed by the preceding vehicle when trying to drive at the desired speed.

This model states that, the maximum speed at which a vehicle (n) can accelerate during a time period ($t, t+T$) is given by:

$$V_a(n, t+T) = V(n, t) + 2.5a(n)T \left(1 - \frac{V(n, t)}{V^*(n)} \right) \sqrt{0.025 + \frac{V(n, t)}{V^*(n)}}$$

where:

$V(n,t)$ is the speed of vehicle n at time t ;
 $V^*(n)$ is the desired speed of the vehicle (n) for current section;
 $a(n)$ is the maximum acceleration for vehicle n ;
 T is the reaction time = updating interval = simulation cycle.

On the other hand, the maximum speed that the same vehicle (n) can reach during the same time interval ($t, t+T$), according to its own characteristics and the limitations imposed by the presence of the leader vehicle is:

$$V_b(n, t+T) = d(n)T + \sqrt{d(n)^2 T^2 - d(n) \left[2\{x(n-1, t) - s(n-1) - x(n, t)\} - V(n, t)T - \frac{V(n-1, t)^2}{d'(n-1)} \right]}$$

where:

$d(n)$ (< 0) is the maximum deceleration desired by vehicle n ;
 $x(n,t)$ is position of vehicle n at time t ;
 $x(n-1,t)$ is position of preceding vehicle ($n-1$) at time t ;
 $s(n-1)$ is the effective length of vehicle ($n-1$);
 $d'(n-1)$ is an estimation of vehicle ($n-1$) desired deceleration.

In any case, the definitive speed for vehicle n during time interval ($t, t+T$) is the minimum of those previously defined speeds:

$$V(n, t+T) = \min \left\{ V_a(n, t+T), V_b(n, t+T) \right\}$$

Then, the position of vehicle n inside the current lane is updated taking this speed into the movement equation:

$$x(n, t+T) = x(n, t) + V(n, t+T)T$$

Calculating the speed of a vehicle on a section

The car-following model is such that a leading vehicle, i.e. a vehicle driving freely, without any vehicle affecting its behaviour, would try to drive to its maximum desired speed. Three parameters are used to calculate the maximum desired speed of a vehicle while driving on a particular section or turning, two are related to the vehicle and one to the section or turning:

1. Maximum desired speed of the vehicle i : $v_{\max}(i)$
2. Speed acceptance of the vehicle i : $\theta(i)$
3. Speed limit of the section or turning s : $S_{\text{limit}}(s)$

The speed limit for a vehicle i on a section or turning s , $S_{\text{limit}}(i, s)$, is calculated as:

$$S_{\text{limit}}(i, s) = S_{\text{limit}}(s) \cdot \theta(i)$$

Then, the maximum desired speed of vehicle i on a section or turning s , $v_{\max}(i, s)$ is calculated as:

$$v_{\max}(i, s) = \text{MIN}[S_{\text{limit}}(i, s), v_{\max}(i)]$$

This maximum desired speed $v_{\max}(i, s)$ is the one referred above, in the Gipps car following model, as $V^*(n)$.

Modelling the influence of adjacent lanes in the car following model

In case that the vehicle is driving along a section, the modified car-following model considers the influence that certain number of vehicles ($Nvehicles$) driving slower in the adjacent right-side lane –or left-side lane, when driving on the left–, may have on the vehicle.

The model calculates first the mean speed for $Nvehicles$ driving downstream of the vehicle in the adjacent slower lane (*MeanSpeedVehiclesDown*). Only vehicles within a certain distance (*MaximumDistance*) from the current vehicle are taken into account.

We distinguish two cases: 1) the adjacent lane is an on-ramp, and 2) the adjacent lane is any other type of lane. Apart from $Nvehicles$ and $MaximumDistance$ parameters, the user can define two additional parameters, $MaximumSpeedDifference$ and $MaximumSpeedDifferenceOnRamp$. Then, the final desired speed of a vehicle at a section is calculated as follows:

```

if (the adjacent slower lane is an On-ramp) {
    MaximumSpeed =  $MeanSpeedVehiclesDown + MaximumSpeedDifferenceOnRamp$ 
}
else {
    MaximumSpeed =  $MeanSpeedVehiclesDown + MaximumSpeedDifference$ 
}
DesiredSpeed = Minimum ( $v_{max}(i, s), \theta(i) * MaximumSpeed$ )

```

This procedure ensures that the differences of speeds between two adjacent lanes will approximately be always lower than $MaximumSpeedDifference$ or $MaximumSpeedDifferenceOnRamp$, depending on the case.

The car-following model in AIMSUN2 has been tested and calibrated using the standard field data that a research group from Robert Bosch GmbH [8]. The test for the car-following model is defined as follows:

“The primary task of a car-following model is to reproduce realistic car-following behavior. The reality has been measured with a radar sensor equipped vehicle recording distance and relative speed to the front car (additionally to the own car’s speed) in a 100 ms cycle, see [9] for further details. One specific sequence of 5 min length has been chosen to perform the comparison. This sequence has been recorded under stop&go conditions during an afternoon peak on a one-lane-per-direction fairly straight road in Stuttgart, Germany. Stop&go is most challenging to the models because the free flow behavior is relatively easy reproducible by any model. During the 5 min sequence several decelerations and accelerations of the front car have been observed. At one moment after 144 sec the front car turned off resulting in a distance step of about 40 m. Because such a maneuver can always happen in real traffic the models have to be able to deal with. Note that it can’t be the target of a simulation model to reproduce exactly the measured behavior of this specific driver in the specific test vehicle. Driver and vehicle variations have to be respected. Hence, the main focus lies on qualitative differences. But the fairly good reproduction of the behavior indicates a model’s realism.

To give an impression of similarity to the measured behavior a quantitative error metric on the distance seems to be reasonable. To avoid overrating discrepancies for large distances a relative metric was chosen weighted by the logarithm and squared. Only the values after each second have been considered”.

The error metric used is: $Em = \sqrt{\sum \left[\log \left(\frac{d_{sim}}{d_{meas}} \right) \right]^2}$ where d_{sim} is the distance of the simulated vehicle,

d_{meas} is the distance measured with the test vehicle, and \log denotes the logarithm base 10. Figure 6 displays the curves for the measured and simulated data. The numerical value for the test error metric for the AIMSUN2 model is 3.4726. These results show that the AIMSUN2 car-following model is able of a fairly good reproduction of the observed values. The numerical value of the error metric outperforms those provided for most of the currently used models (see [8] for details).

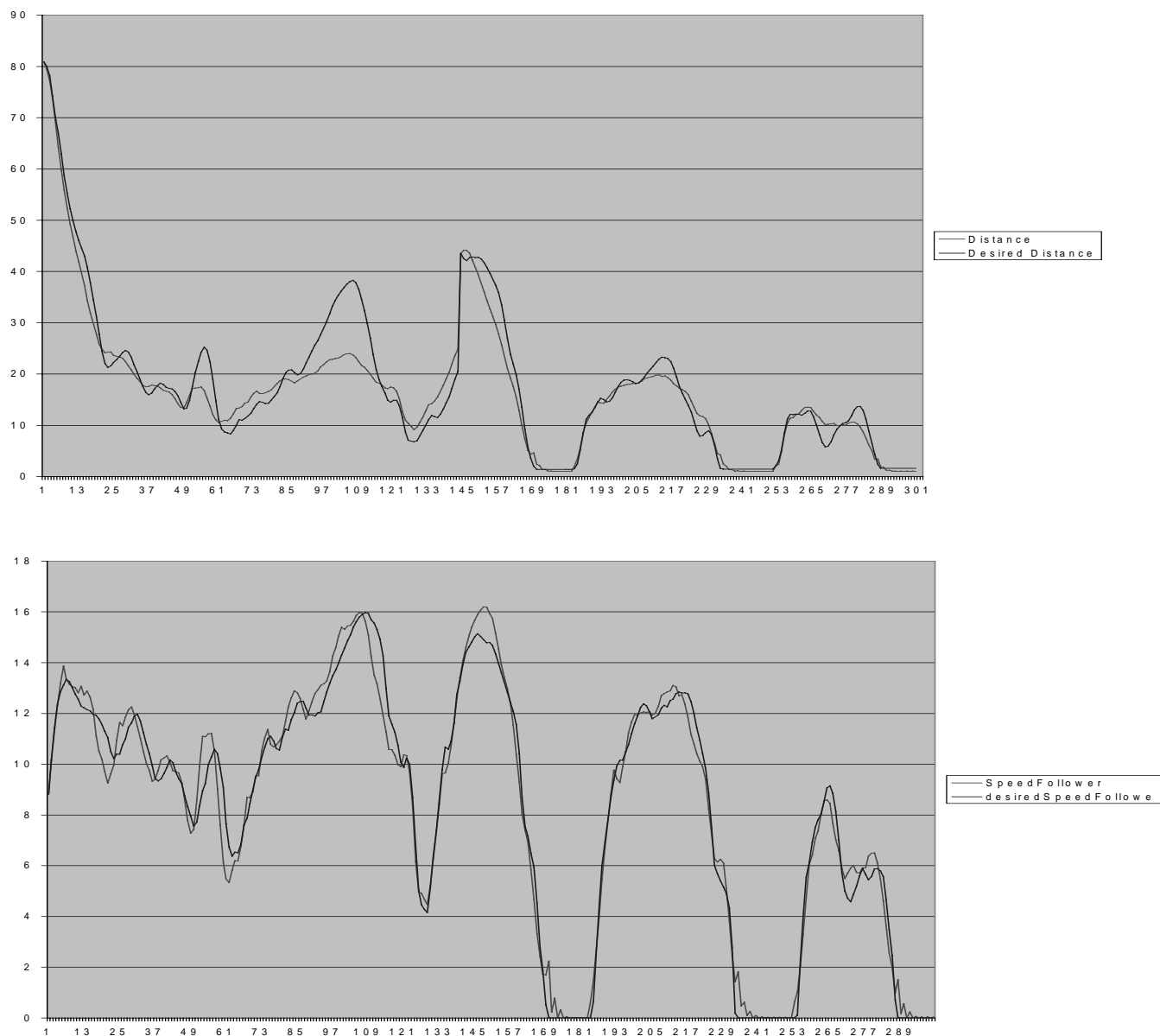


Figure 6: Desired distance and desired follower speed are the simulated values

An additional test to analyse the quality of the microscopic simulator is to check the ability to reproduce macroscopic behaviours. Also the research team at Bosch proposed in [8] a test to compare various microscopic simulators:

“The macroscopic behavior of a microscopic model can be most easily tested by simulating the traffic on cyclic one lane roads. This excludes any effects of lane changes and node passings and concentrates on the car-following task. For this study, a cyclic road of 1000 m length was used. A fixed number of vehicles has been initially set with speed value 0 km/h at randomly determined positions. All vehicles had the same length of 4.5 m and the drivers had the same free flow speed of 54 km/h. Starting with this initial situation a 10 minute time period was simulated without any measurements to reach traffic conditions which are achievable by the model’s behavior itself. After the starting phase the traffic behavior has been recorded at one local measurement point during a simulation time of 2 hours (exact passing time and speed value of each vehicle). The fixed number of vehicles for the simulation run was varied in discrete steps to realize different traffic densities. To visualize the results the traffic flow has been drawn versus the density (given as the number of initially set vehicles on the 1 km ring). The maximal mean traffic flow value of about 1800 veh/h is known as a quite realistic value for longer periods of measurement time. Under urban traffic conditions this maximal flow is typically reached at higher density values than for freeway traffic”.

The results of AIMSUN2 for this second test are displayed in figure 7.

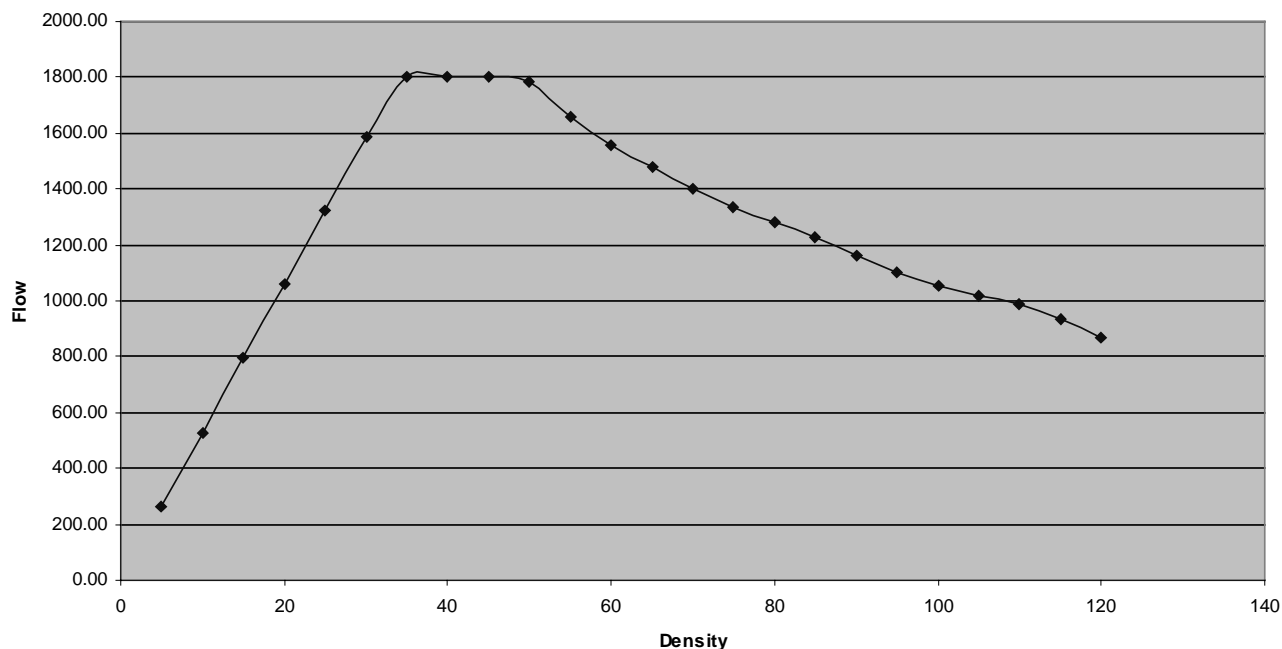


Figure 7: AIMSUN2 flow-density curve for the second test

2.4.2 Lane Changing Model

The lane change model in AIMSUN2 can also be considered as a further evolution of the seminal Gipps lane change model [10]. Lane change is modelled as a decision process analysing the necessity of the lane change (as in the case of turning manoeuvres determined by the route), the desirability of the lane change (as for example to reach the desired speed when the leader vehicle is slower), and the feasibility conditions for the lane change that are also local, depending on the location of the vehicle on the road network. The lane-changing model is a decision model that reproduces approximately the driver's behaviour in the following way:

- Each time a vehicle has to be updated the model draws up the question: *Is it necessary to change lanes?* The answer to this question depends on several factors: the turning feasibility at current lane, the distance to next turning and the traffic conditions in the current lane. The traffic conditions are measured in terms of speed and queue lengths. When a vehicle is driving slower than he wishes, he tries to overtake the preceding vehicle. On the other hand, when he is travelling fast enough, he tends to go back to the slower lane.
- If we answer in the affirmative to the previous question, to succeed in the lane changing we have to answer two more questions:
 - . *Is it desirable to change lanes?*
It means to see if there will be any improvement in the traffic conditions for the driver as a result of the lane changing. This improvement is measured in terms of speed and distance. If the speed in the future lane is faster enough than the current lane or if the queue is shorter enough, then it is desirable to change lanes.
 - . *Is it possible to change lanes?*
It means to verify if there is enough gap to do the lane change with complete safety. For that purpose, we calculate both the braking imposed by the future downstream vehicle to the changing vehicle and the braking imposed by the changing vehicle to the future upstream vehicle. If both braking ratios are acceptable then the lane changing is possible.

In order to achieve a more accurate representation of the driver's behaviour in the lane changing decision process, three different zones inside a section are considered, each one corresponding to a different lane changing

motivation. These zones are characterised by the distance up to the end of the section, which means the next point of turning (see figure 8).

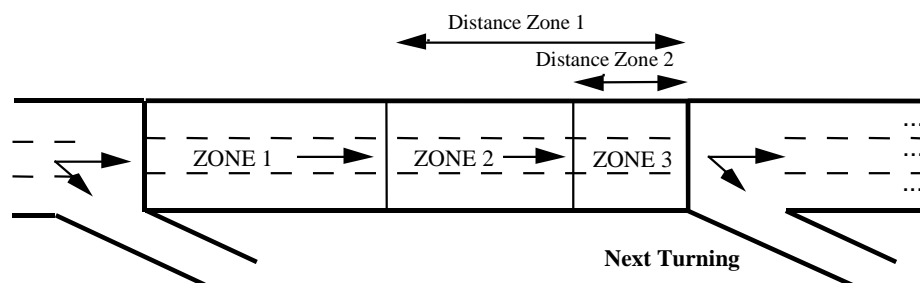


Figure 8: Lane Changing Zones

- Zone 1: It is the farthest from the next turning point. The lane changing decisions are mainly governed by the traffic conditions of the involved lanes; the feasibility of the next desired turning movement is not yet taken into account. To measure the improvement that the driver will get on changing lanes we consider several parameters: desired speed of driver, speed and distance of current preceding vehicle, speed and distance of future preceding vehicle.
- Zone 2: It is the intermediate zone. Mainly it is the desired turning lane that affects the lane changing decision. Vehicles who are not driving on a valid lane (i.e. a lane where the desired turning movement can be done) tend to get closer to the correct side of the road where the turn is allowed. Vehicles look for a gap and may try to adapt to it, but do not affect the behaviour of vehicles in the adjacent lanes.
- Zone 3: It is the nearest to the next turning point. The vehicles are forced to reach their desired turning lanes, reducing the speed if necessary and even coming to a complete stop in order to make the change possible. Also, vehicles in the adjacent lane can modify its behaviour in order to provide a gap big enough for the vehicle to achieve the lane change.

Lane changing zones are defined by two parameters, Distance to Zone 1 and Distance to Zone 2. These parameters are defined in time (seconds) and they are converted into distance whenever it is required for each vehicle at each section using the Vehicle Desired Speed at a Section.

Lane Changing Model at ON-Ramps

A special Lane Changing Model is applied when there is an On-Ramp involved. An additional zone parameter may be defined, *TimeDistanceOnRamp*. This is the Distance, in seconds, from what lateral lanes are considered to be on-ramp lanes, in order to distinguish between a common lateral lane, that is a long lane used for overtaking which drops down, from the proper on-ramp lanes, which are never used for overtaking.

Vehicles driving on a lateral lane that are farther than *TimeDistanceOnRamp* from the end behave as if they were in the Zone 1 of a normal lane. When they are closer than *TimeDistanceOnRamp* to the end of the lane, they behave as having to merge from an on-ramp.

Merge from on-ramp model takes into account whether a vehicles is stopped or not, if it is stopped whether it is at the beginning of the on-ramp queue or not and how long has been waiting. There is another vehicle parameter, *Maximum Waiting Time*, that determines how long a vehicle is willing to wait before getting impatient.

Overtaking Manoeuvre

An overtaking manoeuvre takes place mainly on zone 1, although it can also take place in zones 2 and 3 when the vehicle is in the appropriate turning lane. In order to promote or discourage the overtaking, there are two parameters that the user can define: *PercentOvertake* and *PercentRecover*.

PercentOvertake is the percentage of the desired speed of a vehicle below what the vehicle may decide to overtake. It means that whenever the leader vehicle is driving slower than *PercentOvertake%* of the follower desired speed, the follower vehicle will try to overtake. The default value is 0.90.

PercentRecover is the percentage of the desired speed of a vehicle above what a vehicle may decide to recover the slower lane. It means that whenever the leader vehicle is driving faster than *PercentRecover* % of the follower desired speed, the follower vehicle will try to recover the rightmost (or leftmost) lane. The default value is 0.95.

An overview of the lane-changing model is shown in the diagram of figure 9. The system identifies the type of entity (central lane, off-ramp lane, junction, on-ramp etc.) in which the manoeuvre is going to be done, next determines how the zone modelling should be applied. The current traffic conditions are analysed and the level at which the lane change can be performed is determined, and then the corresponding model is applied.

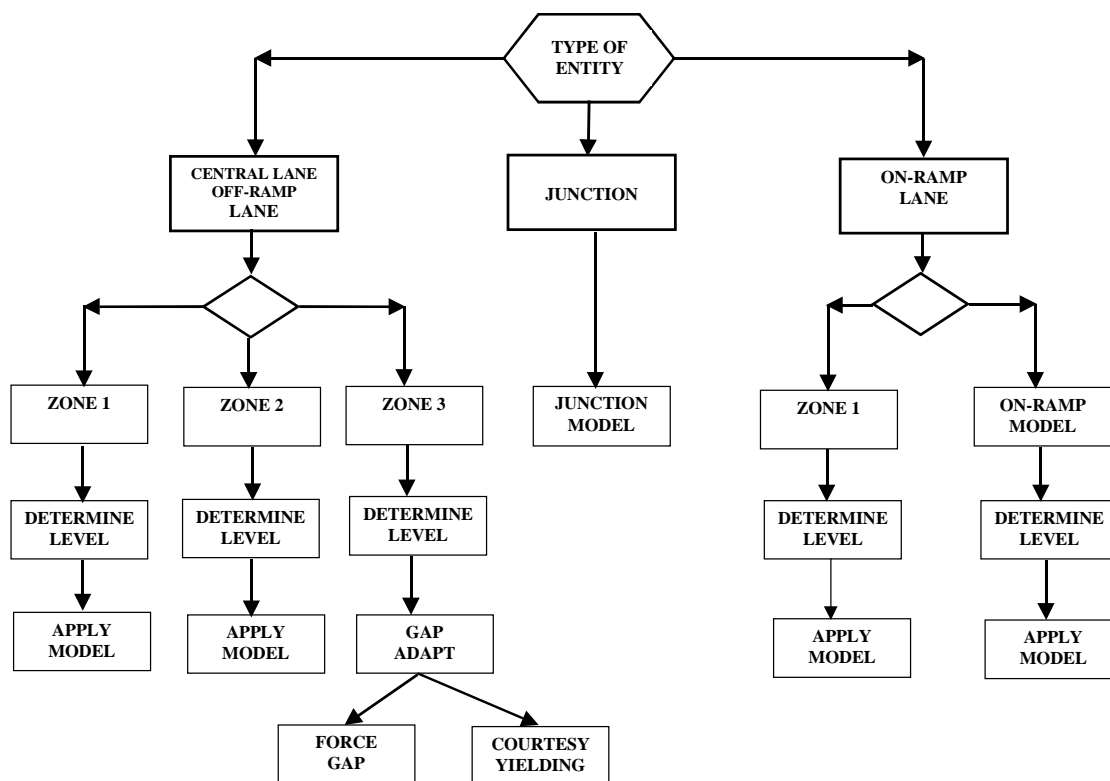


Figure 9

2.5 TEDI network editor

TEDI is a graphical editor for traffic networks (see figure 10). It has been designed with the aim of making the process of network data entry and model building user-friendly. Its main function is the construction of traffic models with which to feed traffic simulators like AIMSUN2. To facilitate this task the editor accepts as a background a graphical description of the network area, so sections and nodes can be built subsequently into the foreground. The editor supports both *urban* and *interurban* roads, which means that the level of detail covers elements such as side lanes, entrance and exit ramps, intersections, traffic lights and ramp metering. TEDI has an interface to the EMME/2 DATA BANK, providing the means to complement a macroscopic analysis effortlessly with a microscopic one using the same traffic data (i.e. O/D matrices). The figure 11 shows the modified version of the conceptual functional structure of GETRAM/AIMSUN2 displaying the interaction between AIMSUN2 and EMME/2. An example on the use of the combination of macro and micro approaches for planning and design of road networks can be found in [10]

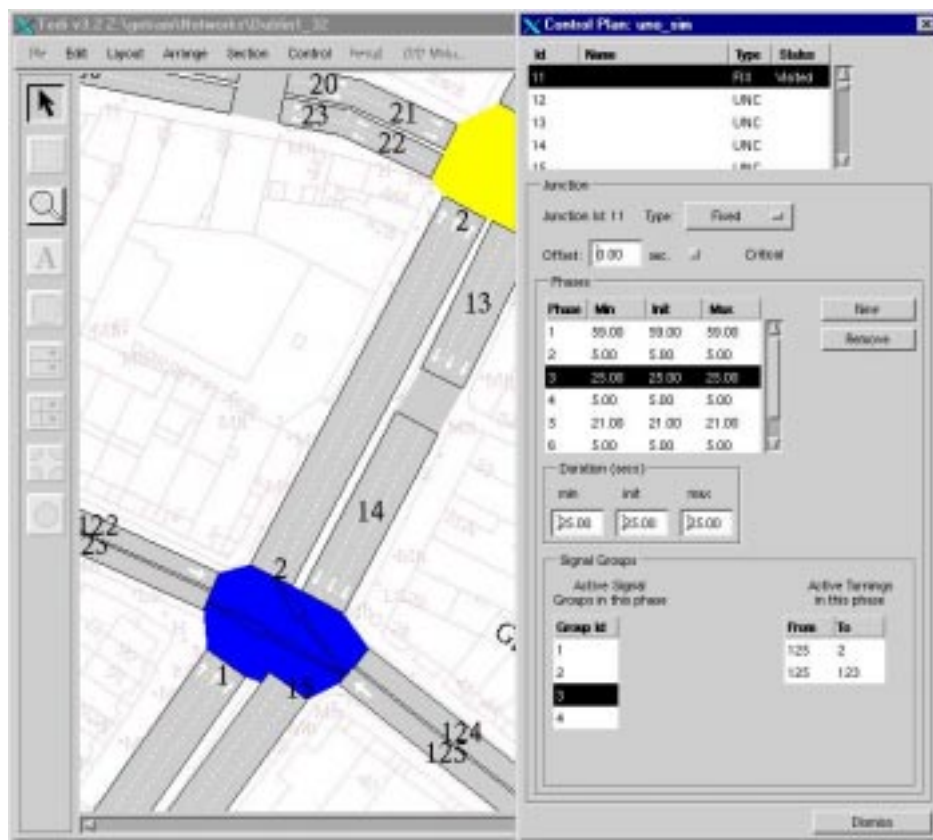


Figure 10: TEDI Graphical Network Editor

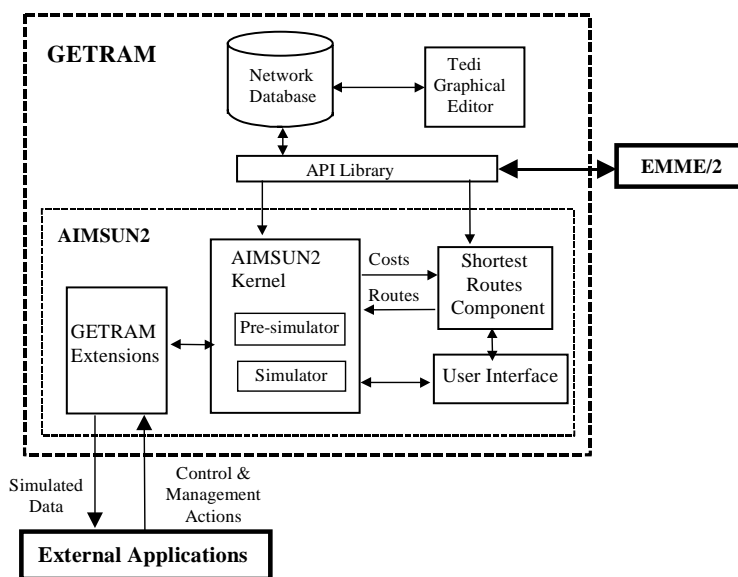


Figure 11

The user can define a hierarchical tree of views so that a traffic model can be restricted to one of these views. The editor is designed for the level of detail required by regional, intermediate and local areas, with increased modelling detail. A library of high-level, object-based application programming functions, named TDFunctions, assists the development of interactive external applications, and, in general to access any data. The TDFunctions enable objects in the network to be read and manipulated or can restrict the view to a sub-area. Results storage and control plans are also accessed with these functions.

3. GETRAM/AIMSUN2 Extensions

To cope with the requirements of simulating Intelligent Transport Systems specific extensions to GETRAM/AIMSUN2 have been developed. These extensions fall into three categories:

- 3.1 Adaptive Traffic Control, Traffic Management Systems and Incident Management Systems
- 3.2 Vehicle Guidance, Fuel Consumption and Emissions
- 3.3 Public Vehicle Monitoring and Control Systems

The approach taken in GETRAM/AIMSUN2 consists of considering the Intelligent Transport System to be tested as an EXTERNAL APPLICATION that can communicate with GETRAM/AIMSUN2. An ad hoc version of AIMSUN2 including a set of DLL has been developed for this purpose. This library gives AIMSUN2 the ability to communicate with almost any of the above-mentioned external applications.

Using the TEDI & AIMSUN2 functions the detector, VMS and traffic lights can be modelled and their attributes defined. The process of information exchange between AIMSUN2 and the external application is shown in Figure 12. The AIMSUN2 model of the road network emulates the detection process providing the external application with the required “*Simulation Detection Data*”. The EXTERNAL APPLICATION (user provided) decides which control and/or management actions have to be applied on the road network and sends the corresponding information to the simulation model which then emulates their operation through the corresponding model components such as traffic lights, VMS, etc. Another set of DLL function enables the user to access the information on each vehicle state (position, speed, acceleration, etc.) at each simulation cycle.

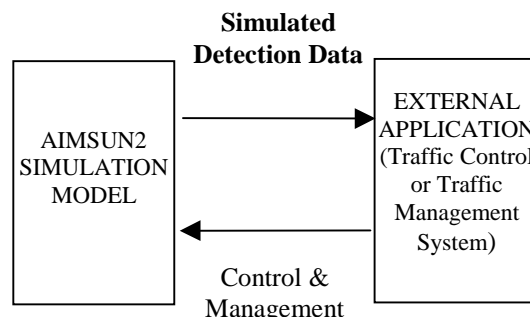


Figure 12

Figure 13 illustrates conceptually how the DLL library of functions works

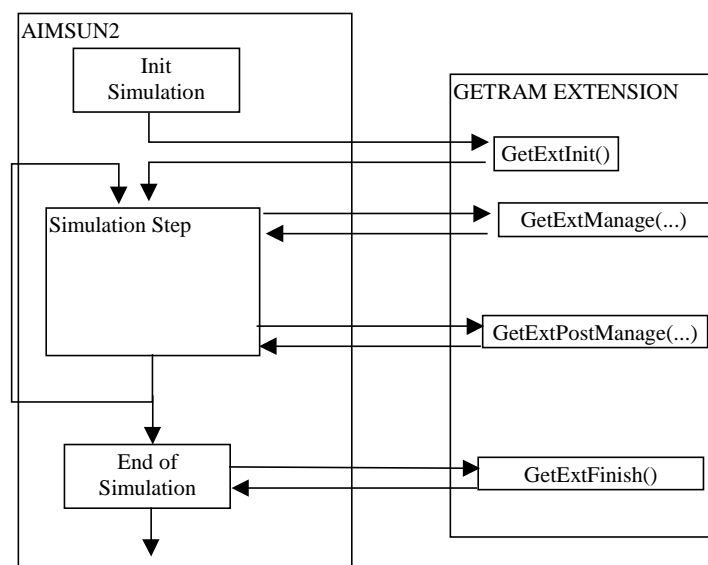


Figure 13

3.1 Simulating Management and Control actions

Three main types of actions, as a result of the actuation of EXTERNAL APPLICATION on the simulation, are taken into account:

- 1) Actuate control of traffic lights and ramp metering
- 2) Actuate control of ramp metering
- 3) Supply information to the driver using Variable Message Signs (VMS)

3.1.1 Traffic Control Actions

AIMSUN2 takes into account two types of traffic control: traffic lights and ramp metering. The first is considered for urban type intersection nodes, while the second one is for controlling freeway entrance ramps.

For the intersection control, a phase-based approach is applied in which the cycle of the junction is divided into phases where each one has a particular set of signal groups with right of way (a signal group is considered as one traffic light). During the simulation of a scenario, AIMSUN2 executes a fixed control plan taking into account the phase modelling for each junction. An EXTERNAL APPLICATION can modify this execution by means of different actions. The available actions are to:

1. Change the duration of each phase: The EXTERNAL APPLICATION can increase or decrease the duration, but the control plan structure is not modified.
2. Disable the fixed control plan structure: The EXTERNAL APPLICATION disables the structure of the control plan and completely controls the phase changing.
3. Change the current phase: The EXTERNAL APPLICATION can change the current phase to another. If the fixed control plan (timings) of the junction have not been disabled, AIMSUN2 programmes the next changing of phase taking into account the duration of the phases. Otherwise AIMSUN2 holds the new phase until the EXTERNAL APPLICATION changes it to another.

Examples of functions of the DLL library relative to control junctions are:

- Read Number of junctions: *Reads the number of junctions present on the road network.*
- Read the Identifier of a junction: *Reads the identifier of junction elem-th present on the road network.*
- Read the Name of a junction: *Reads the name of junction identified by idJunction.*
- Read the number of Signal Groups of a junction: *Reads the number of signal groups defined in junction identified by idJunction.*
- Read the Number of Phases of a junction: *Reads the total number of phases of a junction in the current control plan.*
- Read Time Duration of a phase of a junction: *Reads the maximum, minimum and current duration of a in junction during the current control plan.*
- Read the Current Phase of a junction: *Reads the current phase of a junction, even when the junction has disabled the fixed control plan.*
- Disables the fixed control plan of a junction: *Disables the fixed control plan of a junction, so the phase changing is completely controlled by the EXTERNAL APPLICATION (i.e. an adaptive traffic control system).*
- Change of Phase: *Changes to idPhase in junction identified by idJunction.*
- Change the Current Duration of Phase: *Changes the duration of idPhase in junction identified by idJunction in the current control plan.*
- Change the State of a Signal Group: *Changes the state of signal group idSigGr in junction identified by idJunction.*

An example of this type of application can be found in the final report of the European Project CLAIRE SAVE [11], where an interface with the SCOOT traffic control system was built with the objective of conducting a simulation analysis of the impact of traffic control strategies on fuel consumption and emissions in an urban area.

3.1.2 Ramp Metering

AIMSUN2 also incorporates ramp-metering control. This type of control is used to limit the input flow to certain roads or freeways in order to maintain certain smooth traffic conditions. The objective is to ensure that entrance demand never surpasses the capacity of the main road. AIMSUN2 considers three types of ramp metering depending on the implementation and the parameters that characterise it:

1. Green time metering, with parameters green time and cycle time. It is modelled as a traffic light.
2. Flow metering, with parameters platoon length and flow (veh/h). The meter is automatically regulated in order to permit the entrance of a certain maximum number of vehicles per hour.
3. Delay metering, with parameters mean delay time and its standard deviation. It is used to model the stopped vehicles due to some control facility, such as a toll or a customs checkpoint.

The EXTERNAL APPLICATION can modify this modelling by different actions. It can:

1. Change the parameters of a metering, the EXTERNAL APPLICATION can dynamically modify the parameters that define a ramp metering.
2. Disable the control structure: EXTERNAL APPLICATION disable the structure of the ramp metering and completely controls the state changing.
3. Change the state of a metering: The EXTERNAL APPLICATION can change the current state to another. If the metering has not disabled the control, AIMSUN2 programmes the next changing of state taking into account the parameters, which define the control. Otherwise AIMSUN2 holds the new state until the EXTERNAL APPLICATION changes it to another.

Examples of functions of the DLL relative to ramp metering are:

- Read the Section Identifier of a metering: *Reads the section identifier that contains the metering elem-th present on the road network*
- Read the Type of metering: *Reads the type of metering present in an section. (The type of a metering can be 0: None ;1: GREEN METERING ;2:FLOW METERING*
- Read the Control Parameters of a Green Metering: *Reads the parameters of a green metering that are defined in the current control*
- Change the Control Parameters of a Green Metering: *Changes the parameters of a green metering that are defined in the current control.*
- Read the Control Parameters of a Flow Metering: *Reads the parameters of a flow metering defined in the current control.*
- Change the Control Parameters of a Flow Metering: *Changes the parameters of a flow metering that are defined in the current control.*
- Read the Control Parameters of a Delay Metering: *Reads the parameters of a delay metering that are defined in the current control.*
- Change the Control Parameters of a Delay Metering: *Changes the parameters of a delay metering that are defined in the current control.*
- Disable the fixed control plan of a metering: *Disables the fixed control plan of a metering, so the state changing is completely controlled by the EXTERNAL APPLICATION.;3: DELAY METERING.)*

3.1.3 Variable Message Signs

Providing information to drivers is a possible action of a Traffic Management System on a road network equipped with Variable Message Sign infrastructure. Messages may inform drivers about the presence of incidents, congestion ahead or suggest alternative routes. AIMSUN2 takes into account the modelling of Variable Message Sign (VMS) as defined in TEDI by means of a dialogue including the VMS name or identification code, its position in the section the activated message, if any, the list of feasible messages for this VMS, and the list of all Actions available for this network associated to the messages. Messages in a VMS from its message list may be activated in two different ways: directly through the user interface or by an external application through the communication interface. In both cases, it will cause the message to be displayed as Activated Message and the Actions associated with it to be implemented. Each message has a list of Actions associated with it which appear in the list box named 'Mess Actions' in the VMS Information Window. The list box named 'Actions' contains all actions available for this network. An Action represents the expected impact a message has on driver's behaviour. Examples of Actions are: modifications of the speed limit, modification of the input flow, modifications of the turning proportions. When simulating with the Route Based option actions can also imply a re-routing, that is the

possibility of altering the vehicle's path. This effect is accomplished by defining the next turn and/or defining a new destination. The re-routing effect is defined by the following for each modality independently:

- **Compliance level (δ):** This parameter gives the compliance level of the action. If $\delta=1$ then it causes the re-routing to be followed by all vehicles (i.e. it is obligatory). If $\delta=0$ then the re-routing action will be followed depending on the driver behavioural parameter (a local parameter of it), i.e. it is an information only. When $0 < \delta < 1$, δ gives the level of acceptance, e.g. it is advice
- **Modify the next turning:** Change, which is the next turning that the vehicle must follow. This action is defined taking into account the destination
- **Modify destination:** When a vehicle enters into a section affected by an action, the simulator changes its destination

Functions relative to VMS of the DLL library are:

- Read the Identifier of a VMS: *Reads the section identifier that contains the VMS elem-th present on the road network.*
- Read the message of a VMS: *Reads the text of a message elem-th in a VMS*
- Read the Current Active Message: *Reads the current active message in a VMS*
- Active a Message in a VMS: *Actives in PanelName the Message. AIMSUN2 executes the actions associated in Message*

3.1.4 Detector Measurements

Detection output data is produced by AIMSUN2 periodically, provided that there are detectors defined in the network and the Detection Function of the simulator is activated. Currently there are two main types of Detection implemented: Common Detection Model and EXTERNAL APPLICATION type Detection Model.

In the Common Detection Model, the data produced depends on the measuring capabilities of the detectors. There is a data line for each detector, which contains the detector identifier and the list of measures gathered. They may be Count (number of vehicles per interval), Occupancy (percentage of time the vehicle is on the detector), Speed (mean speed for vehicles crossing the detector) and Presence (if a vehicle has been on the detector, it is set to 1). These data are stored in ASCII files.

In the EXTERNAL APPLICATION type Detection Model, the measures are given at every simulation step or aggregated each detection interval. The gathered measures are: Counts (Number of vehicles), Speeds (Mean speed for vehicles crossing the detector), Occupancies (percentage of time the detector is activated), and Presence (whether a vehicle is over the detector or not). The EXTERNAL APPLICATION can undertake the following actions with detectors: retrieve the number of detectors in the network, retrieve the name of each detector, retrieve the detection interval, retrieve the detector measures gathered in each simulation step, retrieve the aggregated detector measures.

3.1.5 An example of using the GETRAM/EXTENSIONS: Tracking Vehicles

A frequent case when simulating Intelligent Transport Systems is that of having to track individual vehicles, as for instance when a Guidance System is simulated. For dynamic guidance systems based on an exchange of information between the equipped vehicles and the Traffic Information Centre, each equipped vehicle behaves as a floating car generating data on the speed, position, travel time, stops, delays, etc. which relate to the currently experienced traffic conditions and can therefore be used for estimating the guidance information, [12]. The figure 14 displays a snapshot of a simple AIMSUN2 model build to illustrate the use of the GETRAM/EXTENSIONS for simulating an adaptive traffic control system and the tracking of individual vehicles.

The model consists of a ramp with two metering systems, a green time metering (the first), and a delay metering (the second). The snapshot show the queue of vehicles before the first metering when it is red, and a vehicle passing the second metering when it is green. The metering are externally controlled according to the metering parameters. The figure 15 shows the graphics of tracking the speeds of three vehicles, a leader and two followers.

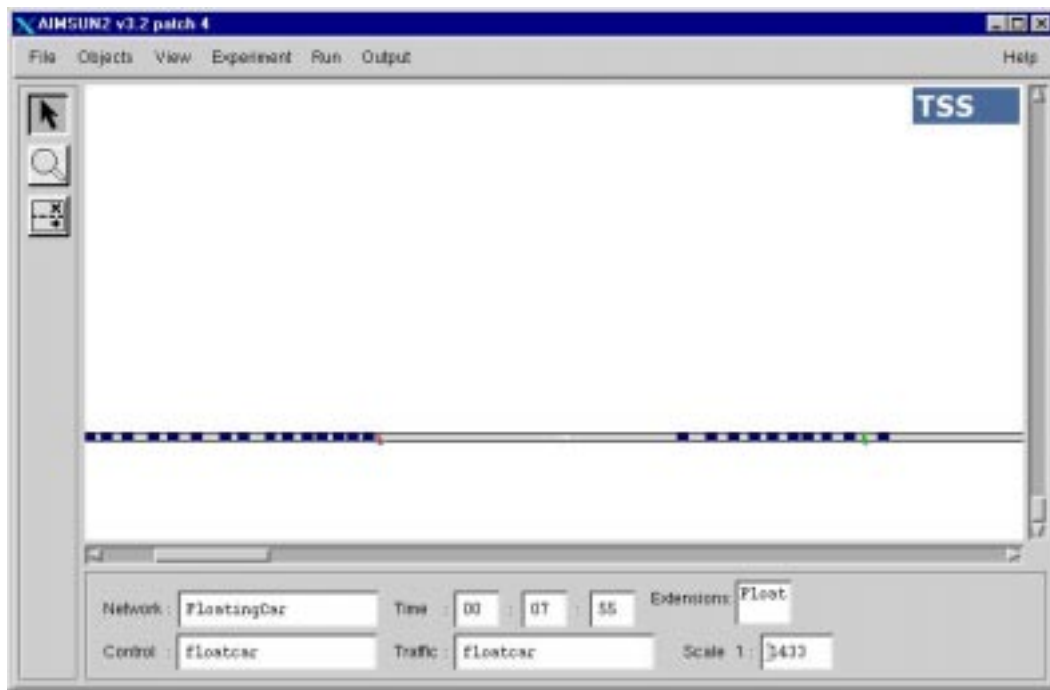


Figure 14. Snapshot of the simple model of the example

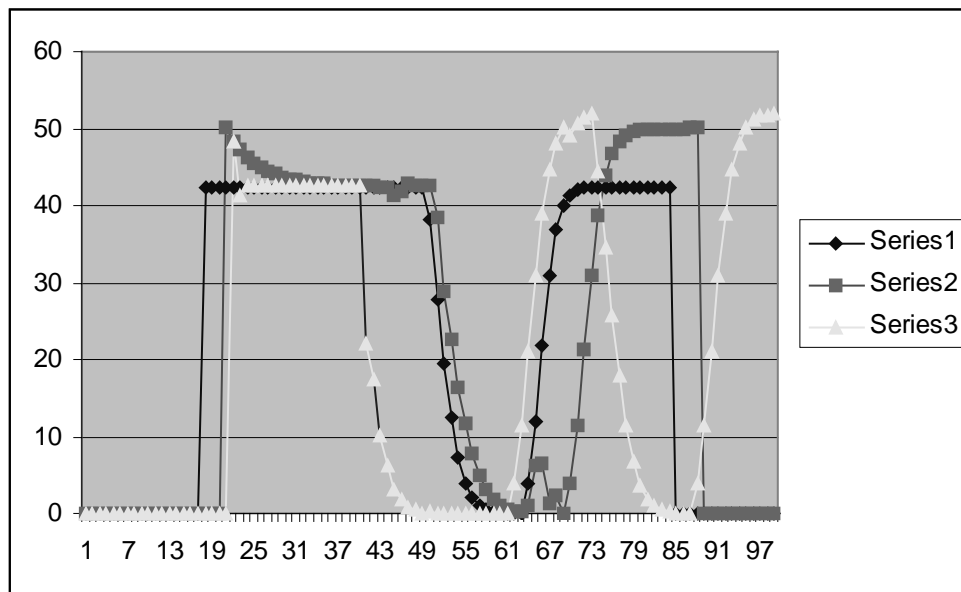


Figure 15: Tracking the speeds of three consecutive vehicles

Series 1 in figure 14 is the graphic representation of the speed of the first vehicle, the leading vehicle, which enters the model at time 15, accelerates until reaching a steady speed, stops at the first metering, resumes the trip once the metering becomes green, accelerates and stops at the second metering. Series 2 and 3 are the graphic representations of speeds of vehicles 2 and 3. The graphic shows how they accelerate, adapt their speeds to that of its respective leader, and the sequence of stops and go produced by the meterings.

3.2 Simulating incidents and incident management

The diagram in figure 16 schematises the methodological procedure proposed for the simulation of incident detection and management based on the EXTERNAL APPLICATIONS. The procedure is based on a microscopic simulation model of a site that emulates traffic conditions on the site, and generates traffic data: flows,

occupancies, speeds, (travel times when required), at the sampling rate requested by the external applications (for example 30 seconds is a standard request for most automatic incident detection algorithms, [13]), with the format proper of the technology used at the site. These traffic data feed the Incident Warning, Incident Detection and Traffic Management Modules implementing the corresponding External Applications.

The Incident Warning applications estimate an incident probability [14] that is sent as a warning to the Traffic Management System that may take it into account. The Simulation model dialogues with the Management System, as an external application, in the way described above. Once the Incident Detection Module detects an incident, it generates an incident alarm, which is sent to the Traffic Management System. The management decisions are communicated to the simulation model through the proper dialogue as described above. Simulation of Vehicle Guidance

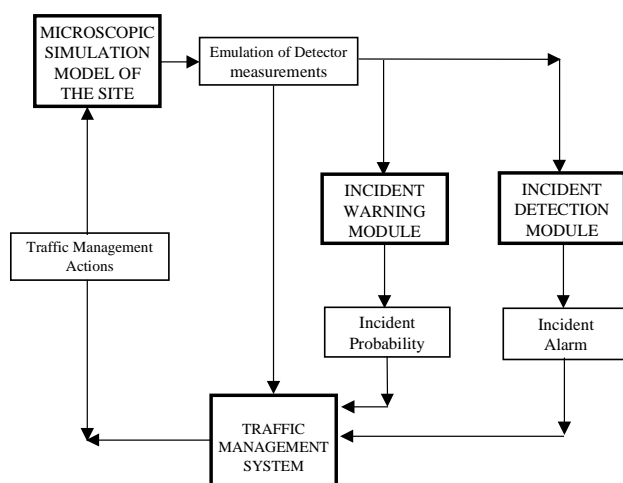


Figure 16

3.3 Simulation of Vehicle Guidance

Vehicle Guidance can also be modelled as an External Application that can be properly simulated with AIMSUN2 by means of an exchange of information using the suitable DLL functions. Essentially the simulation of a Vehicle Guidance system, [15], consists of an exchange of information between the equipped car and the Traffic Information Centre. The on board equipment collects information on the vehicle’s position, travel time, speed, experienced delay, number of stops, etc. which is sent to the Traffic Information Centre, [16-18], by means of the telecommunication technology on which the system is based, as for example, beacons or GSM, or both.

The vehicle data collection can be modelled as follows: each equipped vehicle sends the information to the system when passing through certain points on the network. In the network description the definition of these Data Collection Points (DCP) have been included working like detectors. Each time the position of a guided vehicle is updated it is checked whether or not it has passed through a DCP. If so, a message to the information centre is sent and the vehicle information is updated. Alternative data collection procedures can be implemented for the simulation, as for example assuming that there are a set of fixed DCP which correspond to the end of each section, or that DCP are variable and their position depends on the behaviour of each guided vehicle, for instance, if it takes more than certain time for a vehicle to cross a section or if it has to stop during a section journey. Examples of floating car data collection sampling procedures, as illustrated in figure 17, are: section based (a vehicle sends a message whenever it reaches the end of any section in the network); time based (a vehicle sends a message every certain time interval whose length can be selected by the user as a simulation input parameter); dual mode sampling (time and section based, if a vehicle takes more than one time interval to travel one section, a message is sent for each time interval); speed and section basis: a vehicle sends a message at the end of each section. Besides, if during the trip along a section the vehicle speed falls below a minimum value (which is set by the user as a simulation input parameter), a message is sent every time interval until the end of section is reached.

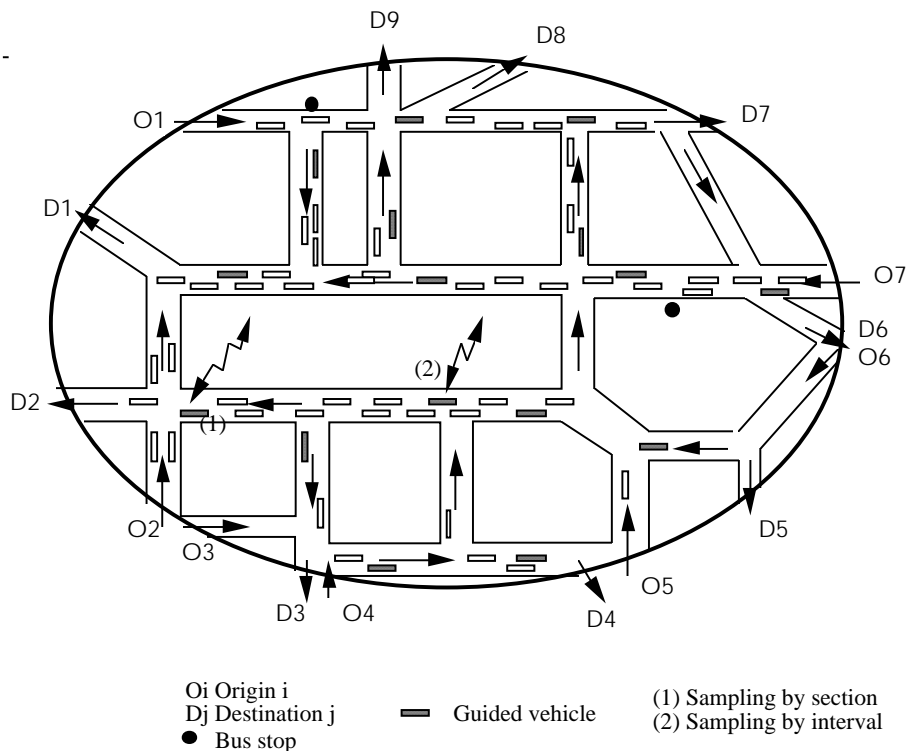


Figure 17: Microscopic Simulation of Vehicle Guidance Systems

Guided vehicles, which are a percentage of the total, are identified at generation time and the data collection process is initialised. For each guided vehicle the corresponding DLL functions collect the information on the vehicle data. Additionally, each time a guided vehicle is updated, it is checked whether or not it is time to transmit data. If so, the following information is updated by means of the ad hoc DLL: Number of messages sent, Number of data blocks transmitted, Sections and distance travelled since last message, Time for the next transmission. In a similar way every time a guided vehicle reaches the end of a section, the following information is updated: Number of messages sent, Number of data blocks transmitted, Distance travelled since last message, Time since last transmission.

The simulation can thus provide the following information [12]: traffic volumes (guided and unguided); mean vehicles speed (km/h); number of stops per vehicle/kilometer; communications overhead in total number of messages between vehicles and information center, per time unit, and total number of blocks transmitted per time unit; mean time between messages per vehicle; average distance travelled between messages per vehicle and average number of messages per section and vehicle. The first three items estimate the level of network congestion in the simulation experiment. The next two estimate the communications requirements for the system, and the last three are measures about the quality of the overall information received by the centre. To illustrate these simulation results let us summarily describe a typical simulation experiment of a guidance system. The parameters defining the experiment are the following: **MNV**, mean number of vehicles in the network; **MNG**, mean number of guided vehicles in the network; **NTR**, number of trips / hour. Trips from origins to destinations in the modeled network; **SPD**, mean journey speed in kilometers/hour; **NST**, number of stops per vehicle per kilometer traveled. The results on communications requirements produced as output by the AIMSUN2 simulation are represented by the following variables: **MpS**, number of messages (transmissions) per second sent from guided vehicles to the information center; **BpS**, number of data blocks transmitted per second. For instance, a trip message is composed by one data block for the header and one data block for each section traveled. Taking into account the type of information of each section of the messages, the typical length of a data block could be considered as 16 bytes. AIMSUN2 output provides also an estimate of the quality of the overall data transmitted from the equipped vehicles to the information centre by means of the following variables: **TbT**, mean time between transmissions per vehicle, which is a measure of how often the center has information from a given vehicle; **DbT**, mean distance travelled between transmissions per vehicle, this measure together with the **TbT**, gives an idea about the frequency of information updating for each vehicle; **CpS**, mean number of messages transmitted per section per vehicle. This provides measures of the information quality for each section.

Thus, for example, for a simulation experiment done in the SOCRATES project [12], [16-17], the traffic conditions were characterised by the following values:

MNV	MNG	NTR	SPD	NST
10000 veh	500 veh	54000 trips	30 km/h	6.2 stops

Therefore, the mean number of guided vehicles that the system was controlling was 500 vehicles from a total of 10000 (that is the 5%). The mean speed on the network was 30 km/h and a vehicle had to stop on average about 6 times every kilometre travelled.

The communications requirements obtained in that simulation experiment for the different sampling procedures are summarised in the following table:

	section	time30	time60	dual30	dual45	dual60	speed30	speed45	speed60
MpS	10.3	16.3	8.0	17.5	13.7	12.0	19.7	17.6	16.7
BpS	20.6	42.2	25.3	42.6	31.2	26.1	49.2	42.9	40.2

The quality of the data received by the traffic information centre can be represented by the following measures:

	section	time30	time60	dual30	dual45	dual60	speed30	speed45	speed60
TbT	44.5"	30.0"	60.0"	21.3"	27.0"	30.8"	18.9"	21.1"	22.2"
DbT	299 m	185 m	366 m	171 m	217 m	248 m	152 m	170 m	180 m
CpS	1	1.57	0.77	1.76	1.38	1.21	1.98	1.77	1.68

From this experiment, it appears that the best sampling procedures are those based on dual mode sampling on time basis. For not to fall in communication overheads, a time interval of 60 sec. or more is preferable. The Traffic Information Centre collects the individual information from the equipped vehicles and, after a suitable processing produces the guidance information which is transmitted to the guided vehicles. The broadcasting of this information and the guided vehicle reactions can be simulated using the DLL in a similar way as the simulation of the VMS, based on the capability of the simulator to dynamically re-route the guided vehicles en-route.

4. The parallelization of AIMSUN2

To conclude this description of improved modelling features in the microscopic simulator AIMSUN2 we will make some comments on what could be expected from the parallelisation of the simulator. A complete description can be found elsewhere, [19-21]. Some reasons to parallelize a microscopic traffic simulator could be the following: the situation of the current practices in traffic management can be summarised saying that, out of a limited traffic control practice, traffic management is currently based on manual procedures, relying on the experience of human operators, and off-line computing practices. There is a main reason for that, numerical algorithms, based either on optimisation or simulation approaches, to deal with time-varying traffic flows in real size traffic networks have very high computing requirements to be processed sequentially on the currently available computing platforms, the resort to parallel computing is a way for achieving the required performance for real-time applications. The encouraging results obtained with the parallel version of AIMSUN2, reported in [21], based on a standard library of threads available in the current version of SUN Solaris 2.4, and a shared memory computing platform, where average speed ups of up to 3.5 times have been achieved, show that a quasi-real time operation of the systems using microscopic simulation for management purposes is feasible. In the computational experiments only the basic version of AIMSUN2 has been parallelized, certainly the inclusion of routing information and its use in the microsimulation will impose additional computational burden that must be investigated. On the other hand the parallelization studied depends, obviously, on the structure of AIMSUN2 and therefore the results cannot be extrapolated to other microscopic simulator with different internal structures. However, we believe that our results show that the parallelization of the microscopic simulators, on the currently available computer platforms, opens the door to simulation analysis of medium to large networks, and not only small networks as microscopic approaches had been restricted so far, and to the use of simulation as decision support tool in the context of Traffic Management applications

5. Conclusions

The paper has described the requirements to simulate Advanced Transport Applications and the conceptual approaches to implement them in microscopic traffic simulators. The description has been completed showing how these approaches have been implemented in the case of the AIMSUN2 microscopic traffic simulator. These implementations have been mainly done through the participation in projects of the ATT Programme of the European Union. References to these implementations can be found in the already referenced reports of SMARTTEST (simulation of VMS, ramp metering, etc.) [1], SOCRATES (Vehicle Guidance) [16-17], CLAIRE SAVE (Adaptive Traffic Control and Environmental impacts) [11], CAPITALS (Traffic Management) [22], IN-RESPONSE (Incident Management) [23], and PETRI, [20].

6. References

1. SMARTTEST Project Deliverable D3, August 1997, European Commission, 4th Framework Programme, Transport RTD Programme, Contract N^o: RO-97-SC.1059.
2. R. Grau and J. Barceló. *The design of GETRAM: A Generic Environment for Traffic Analysis and Modeling*. Research Report DR 93/02. Departamento de Estadística e Investigación Operativa. Universidad Politécnica de Cataluña. (1993).
3. J. Barceló, J.L. Ferrer and R. Grau. *AIMSUN2 and the GETRAM Simulation Environment*. Technical Report. Departamento de Estadística e Investigación Operativa. Universidad Politécnica de Cataluña (1994).
4. INRO Consultants, *EMME/2 User's Manual*. Software Release 8.0 (1996).
5. J. Barceló and J.L. Ferrer. *AIMSUN2: Advanced Interactive Microscopic Simulator for Urban Networks. User's Manual*. Departamento de Estadística e Investigación Operativa. Facultad de Informática. Universidad Politécnica de Cataluña (1997).
6. P.G. Gipps, *A Behavioural Car-following Model for Computer Simulation*, Transpn. Res. 15B, pp 105-111 (1981).
7. D. Manstetten, W. Krautter and T. Schwab, *Traffic Simulation Supporting Urban Control System Development*, Robert Bosch GmbH, Corporate Research and Development, Information and Systems Technology, P.O. Box 10 60 50, 70049 Stuttgart, Germany, (1998).
8. T. Bleile, W. Krautter, D. Manstetten and T. Schwab, *Traffic Simulation at Robert Bosch GmbH*, Proc. Euromotor Seminar Telematic / Vehicle and Environment, Aachen, Germany, Nov. 11-12, (1996)
9. P.G. Gipps, *A Model for the Structure of Lane-Changing Decisions*, Transpn. Res. 20B, pp.403-414 (1986).
10. L. Montero, E. Codina, J. Barceló and P. Barceló, *Combining Macroscopic and Microscopic Approaches for Transportation Planning and Design of Road Networks*, Proceedings of the 19th ARRB Meeting, Sydney, (1998).
11. K. Fox, H. Kirby, G. Scemama and A. Walker, *CLAIRE SAVE: Avoiding High Fuel Consumption in Congested Cities*, Project Final Report, Institute for Transport Studies, University of Leeds, (1996)
12. J. Barceló, J.L. Ferrer, and R. Martín, *Simulation Assisted design and Assessment of Vehicle Guidance Systems*, Accepted for publication in International Transactions on Operations Research, (1998).
13. Y.J. Stephanedes, A.P. Chassiakos and P.G. Michalopoulos, *Comparative Performance Evaluation of Incident Detection Algorithms*, Transportation Research Record 1360 (1996).
14. I.R. Wilmlink and L.H. Immers, *Deriving Incident Management Measures using Incident Probability Models and Simulation*, TNO Research Report 95/NV/172, The Netherlands (1995).
15. D. Jeffery. *Route guidance and In-vehicle Information Systems*. In: *Information Technology. Applications in Transport*, P. Bonsall and M. Bell (eds), VNU Science Press, Utrecht, pp. 319-351 (1987).

16. SOCRATES, 1990, DRIVE I Project V1007, Commission of the European Communities, Report on WP1.2.3, 'Floating Car Data', Prepared by Hoffmann Leiter, Universitat Politècnica de Catalunya, and BAST, responsables G.Hoffmann and J.Barceló.
17. SOCRATES KERNEL 1992, DRIVE II Project V2013, Commission of the European Communities, Report SCKN/UPC 03.43.92, *Equipped Vehicle Fleet Requirements for Monitoring Network Conditions in a Dynamic Route Guidance System* Prepared by Universitat Politècnica de Catalunya, responsible J.Barceló.
18. I.Catling, "SOCRATES", in: *Advanced Technology for Road Transport*, ed. By Ian Catling, Artech House, London (1994).
19. J.Barceló, J.L. Ferrer, D. García, M. Florian and E. Le Saux, The Parallelization of AIMSUN2 Microscopic Simulator for ITS Applications, Proceedings of the 3rd. World Congress on Intelligent Transport Systems, Orlando (1996).
20. J.Barceló, J.Casas, E.Codina, A.Fernández, J.L.Ferrer, D. García and R.Grau, PETRI: A Parallel Environment for a Real-Time Traffic Management and Information System, Proceedings of the 3rd. World Congress on Intelligent Transport Systems, Orlando, (1996).
21. J. Barceló, J.L. Ferrer, D. García, M. Florian and E. Le Saux, "Parallelization of Microscopic Traffic Simulation for ATT Systems Analysis", in: *Equilibrium and Advanced Transport Modelling*, ed. By P. Marcotte and S. Nguyen, Kluwer (1998).
22. CAPITALS, (1998), "Workpackage 5.2 Evaluation Report", EU DGXIII, Project TR 1007.
23. IN-RESPONSE, (1996, 1997) Deliverables 4.3, 5.1 and 5.2, EU DGXIII, Project TR1030.