3 Dynamic Traffic Assignment

3.1 Introduction

Traffic simulation models have proven to be suitable tools for the evaluation of intelligent transport systems (ITS), such as advanced traffic management systems and adaptive traffic control systems. The applicability of traffic simulation strongly depends, in most cases, on computing performance. Off-line applications, such as those employed in testing ITS strategies and in planning studies, are best served by fast-running traffic simulation models because of their iterative nature. Models that are faster than real time are a requisite for on-line applications, such as those that support decision-making in real-time traffic management. A key aspect in the evaluation of ITS strategies is the impact on drivers' behaviour, namely with respect to path choice. The dynamic traffic assignment (DTA) model emulates drivers' decision-making processes.

From an analytical point of view, dynamic traffic assignment has usually been associated with dynamic user equilibrium problems. Some of the most successful approaches to these problems have been inspired by the seminal paper by Friesz et al. (1993), which proposes a dynamic network user equilibrium model that equilibrates the disutilities of temporal choices. To achieve such equilibrium, they assume "that the essential choices available to users of a transport network—route choice and departure time—occur in time-varying environments and are made rationally" and they further conclude that these rational choices can only be made if the disutilities of the alternatives are equilibrated.

Two main approaches were used to model these route choices. The first was based on a generalization of Wardrop's first principle of static traffic assignment, in which users try to optimize their route based on the information available to them. This approach describes the evolution of flows when users make route-choice decisions based on the travel times experienced and is usually known as a preventive or en-route assignment. It does not achieve a day-to-day equilibrium pattern and is therefore considered a dynamic traffic assignment principle and not a true equilibrium. In the aforementioned paper, Friesz et al. propose an alternative generalisation of Wardrop's principle that is 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 a 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 travellers combine the travel times experienced with conjectures in order to forecast temporal variations in flows and travel costs.

In the aforementioned paper, Friesz et al. formulated the dynamic equilibrium problem in the space of path flows $h_k(t)$, for all paths $k \in K_i$, as the set of feasible paths for the *i*-th OD pair at time *t*, where time index *t* refers to the period when the flow departures from its origin. The path flow rates in the feasible region Ω satisfy, at any time $t \in (0,T)$, the following flow conservation and non-negativity constraints:

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

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 time interval *t*. The approach assumes that optimal user equilibrium conditions can be defined as "*a temporal version of the static Wardrop user-optimal equilibrium conditions"* (Friesz et al., 1993), which can be formulated as follows:

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

$$u(t) = \underset{k \in K_i}{Min} \{ s_k(t) \}$$

where $s_k(t)$ is the path travel time on path *k*, which is determined by dynamic network loading. Friesz et al. (1993) show that these conditions are equivalent to the variational inequality problem, in vector form, which consists in finding $h^* \in \Omega$, such that

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

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

- A method for determining path-dependent flow rates on the paths in the network.
- A dynamic network loading method, which determines how these path flows give rise to time-dependent arc volumes, arc travel times and path travel times.

The diagram in Figure 3.1 depicts the logical, conceptual approach to dynamic traffic assignment models. Path flow rates depend on the emulation of drivers' path choice behaviour, determining in this way which of the alternative approach is implemented:

- Dynamic en-route assignment. At each time period, the corresponding fraction of the demand is assigned to the currently available paths for each origin-destination pair according to the probabilities estimated by a route choice model. Drivers can be allowed to dynamically change route en route if a better path from their current position to their destination becomes available.

- Dynamic equilibrium assignment. Path flows are determined by an approximate solution to the mathematical model for dynamic equilibrium conditions (Florian et al., 2001), (Barceló et al., 2002).

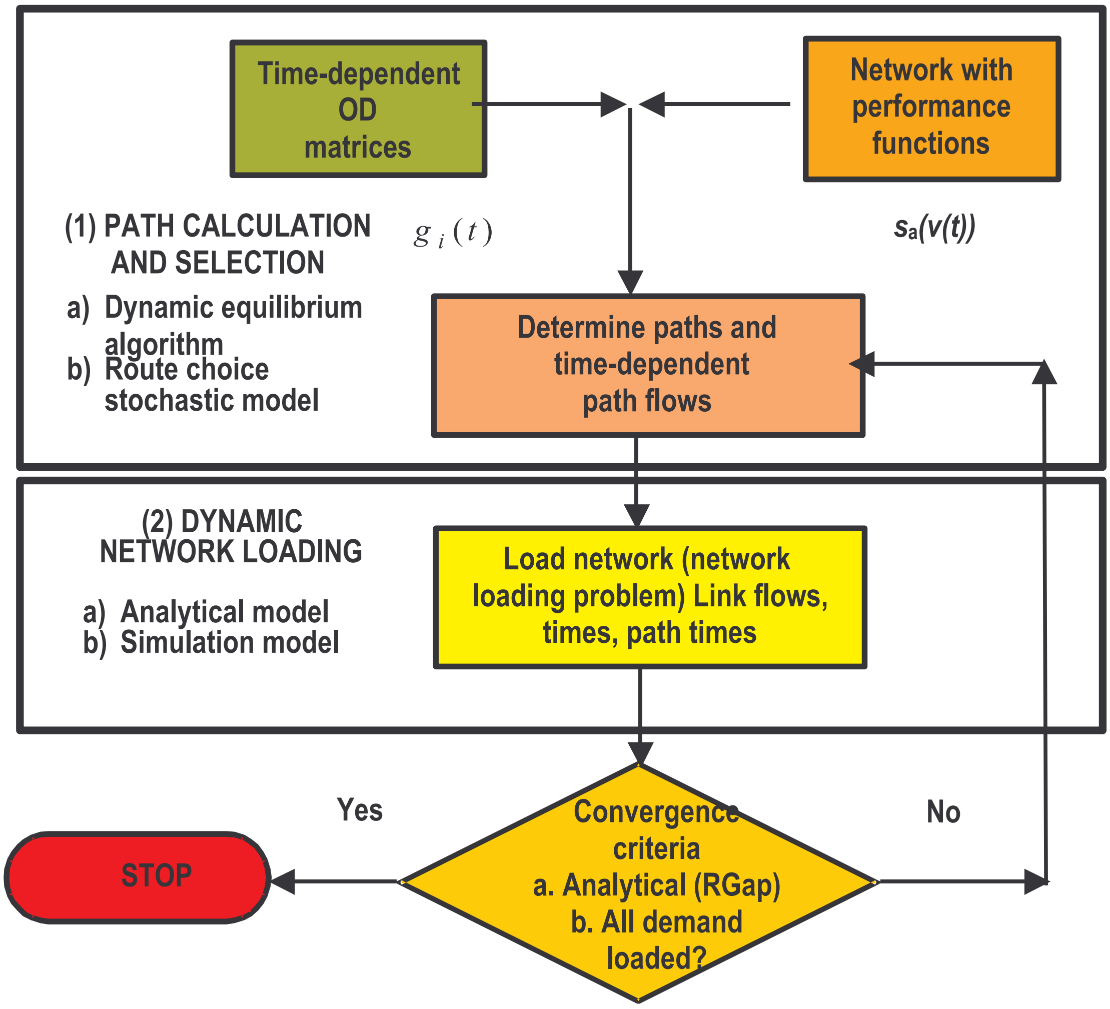

Time-dependent OD matrices
Network with performance functions
(1) PATH CALCULATION AND SELECTION
a) Dynamic equilibrium algorithm
b) Route choice stochastic model
$g_i(t)$
$s_a(v(t))$
Determine paths and time-dependent path flows
(2) DYNAMIC NETWORK LOADING
a) Analytical model
b) Simulation model
Load network (network loading problem) Link flows, times, path times
Yes
No
STOP
Convergence criteria
a. Analytical (RGap)
b. All demand loaded?

Figure 3.1. Conceptual approach to dynamic traffic assignment

On the subject of dynamic network loading, which is also known as dynamic network flow propagation Cascetta (2001) states that "*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 have been proposed, from the analytical approaches of Wu (1991), Wu et al. (1998a, 1998b) and Xu et al. (1998, 1999) to the simulation-based approach of Florian et al. (2001). This work explores the feasibility of a dynamic network loading mechanism based on the AIMSUN model (Barceló et al., 1995, 1998, 2002), (Codina and Barceló, 1995). This traffic simulation approach is proposed not only because of its ability to capture the full dynamics of time-dependent traffic phenomena but also for being capable of dealing with behavioural models that account for drivers' behaviour in the way that is required in dynamic network loading.

However, the simulation of such systems 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 models, vehicles have neither origins nor destinations and move randomly about the network. The simulation approach required should be based on a new microscopic simulation paradigm: a route-based microscopic simulation. In this approach, vehicles are input into the network according to the demand data defined as an OD matrix (preferably time-dependent) and they drive around the network following specific paths in order to reach their destination. In the main route-based simulation process, new routes are to be calculated periodically during the simulation and a route choice model is needed when alternative routes are available.

3.2 DYNAMIC TRAFFIC ASSIGNMENT IN AIMSUN

To conduct research on how microscopic simulation can achieve a dynamic traffic assignment according to the formal approaches described in the previous section, we needed a version of the AIMSUN simulator that was able to perform the functions required. Therefore, our first task was to develop and implement these new functionalities in AIMSUN, a fact that should also be considered one of the main contributions of this thesis, insofar as these new functionalities have become standard components of the new commercial versions of AIMSUN (AIMSUN, 2002) and have thus been made available to traffic analysts. For a better understanding of how these new features work, Appendix V provides a summary description of how AIMSUN models the network and the traffic demand descriptions required for dynamic traffic assignment and Appendix VI provides a summary description of all the new developments implemented to enable dynamic traffic assignment to be conducted following the approaches described in the previous section.

For dynamic traffic assignment, traffic demand is defined in terms of an OD matrix, which gives the number of trips from every origin centroid to every destination centroid, for each time slice, for each vehicle type. When a vehicle is generated at an origin, it is assigned to one of the available paths and the vehicle's origin is thus connected to the destination. The vehicle will travel along this path until it reaches its destination unless it is allowed to dynamically change route en route (i.e. guided vehicles) when there is a better route, from its current position on the path assigned, to its destination. The simulation process based on time-dependent paths consists in carrying out the following steps:

Step 0. Calculate the initial shortest path(s) for each OD pair using the initial costs defined.

Step 1. Simulate for a predefined time interval (e.g. 5 minutes) and assign to the paths available the fraction of the trips between each OD pair for that time interval according to the selected route choice model. Obtain new average link travel times as a result of the simulation.

Step 2. Taking into account the average link travel times, recalculate the shortest path.

Step 3. If there are guided vehicles or variable message signs that suggest rerouting, provide the drivers who are allowed to dynamically reroute during a trip with the information calculated in Step 2.

Step 4. Go to Step 1.

Repeat until all the demand has been assigned in place of the convergence criteria shown in Figure 3.1, which are used by the analytical model to control the length of the simulation.

The following are the two main tasks that must be carried out if the dynamic traffic assignment procedure is to be modelled:

- Paths (or routes) from origins to destinations must be computed at each time interval.
- Path selection (or assignment) is the process that is applied to each vehicle, either when it enters the system or during its trip. The process consists in selecting one path or route from all the paths available for it to reach its destination.

3.2.1 PATH DEFINITION

The paths available from an origin to a destination, which are taken into account in the path selection process for the vehicle, can either be defined by the user (user-defined paths) or calculated by applying a shortest path algorithm, which uses the concept of “cost”. In Section 3.2.1.2 , this concept, and how it has been designed and implemented in this work, is explained in more detail.

- User-defined paths. These paths correspond to the idea of well-known paths, or the most familiar paths for drivers, from an origin to a destination according to the analyst’s knowledge of the network modelled. They are predefined by the analyst that is using the network editor or taken as an output from other traffic simulators or transport models, whether macroscopic (i.e. in transport planning) or microscopic.
- Shortest paths calculated. These are calculated by applying the shortest path algorithm to a network representation in terms of links and nodes, in which each link has an

associated cost function (i.e. travel time) that will be used in the shortest path calculation.

3.2.1.1 NETWORK REPRESENTATION

In terms of sections and intersections, the network representation used for the microscopic simulation is inappropriate for the shortest path computation procedure, which requires a link-node representation. To explicitly account for turning movements in the translation from the AIMSUN representation to the link-node representation, a link that connects two nodes models both a section and a turning movement. Therefore, each AIMSUN section is split into as many links as there are turning movements. The computation of the shortest paths uses a label-setting method, in which the labels are associated with a link, which means that different costs can be assigned to each turning of a section.

Figure 3.2 depicts an example of an AIMSUN network composed of sections, junctions and joins. The corresponding network representation used by the shortest route component, which is composed of nodes and links, is shown in Figure 3.3. Note that, for each section, a node is created and there is a link for each turning movement. The cost assigned to each arc is a function of the travel time of the section plus the travel time of the turning movement.

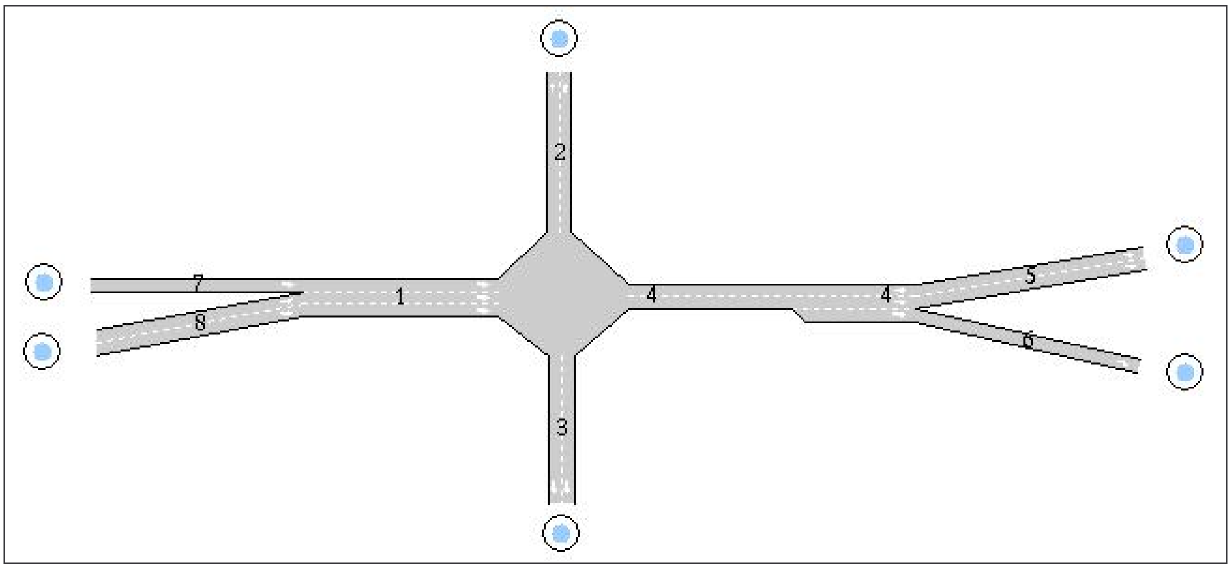


Figure 3.2. Example of an AIMSUN network

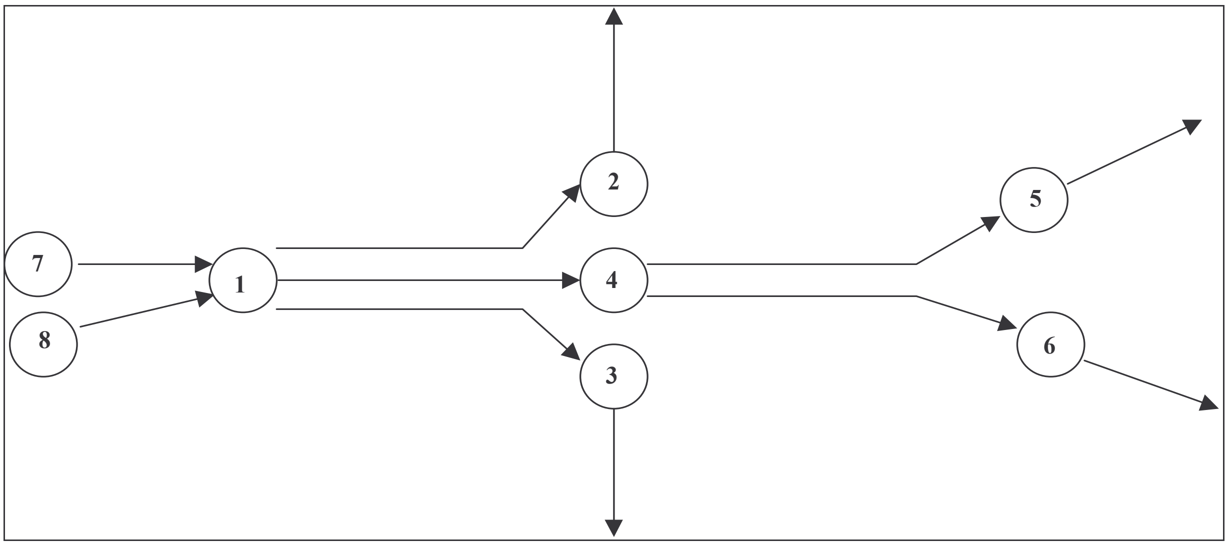


Figure 3.3. Representation of the network in Figure 3.2 for the shortest path calculation

3.2.1.2 LINK COST FUNCTIONS

In the link-node representation of the network used in calculating the shortest routes, each link is associated with a cost function. Two types of link cost functions are used to calculate the shortest path, depending on whether simulated data (i.e. simulated travel time) are available or not, and they are the *initial cost function* and the *dynamic cost function*. In both cases, there is a default cost function, which represents link travel time in seconds (a section travel time plus the turning movement travel time; this last term introduces the penalty of the turning movement, if it exists).

3.2.1.2.1 LINK CAPACITY

The numerical attribute of a link is its theoretical capacity, which can also be used as an argument in the cost function. This capacity is calculated using the theoretical user-defined section capacity of section *s* to which the link belongs. In the calculation of the link capacity, the number of lanes in the section that are used for the turning movement that corresponds to each link and the number of turning movements shared by each lane are considered.

The weight of a given lane *i* is defined as

$$WL_i = \begin{cases} 1 & \text{, if i}^{\text{th}} \text{ lane is a central lane} \\ L_i / L_s & \text{, if i}^{\text{th}} \text{ lane is an exit or side lane} \end{cases}$$

where L_i is the length of lane *i* and L_s is the length of section *s*, which contains lane *i*.

The weight of a lane denotes the lane's contribution in the calculation of the link capacity. A central lane makes a complete contribution to the capacity link, that is, it has a weight of 1, whereas the contribution of an exit or side lane is proportional to its length compared to the length of the section to which belongs. In Figure 3.4, the weight of the side lane is 0.2 (10/50) and in Figure 3.5 the weight of the side lane is 0.8 (40/50).

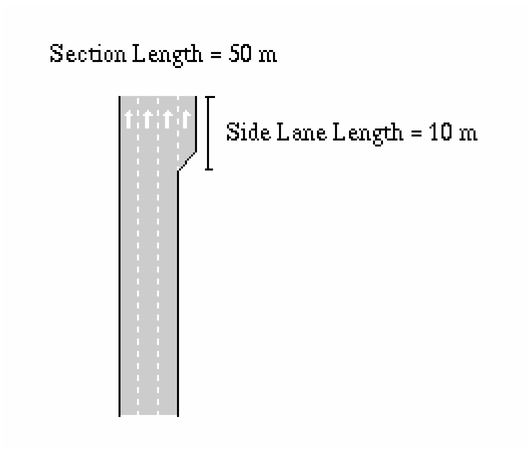


Figure 3.4. Example of a short side lane

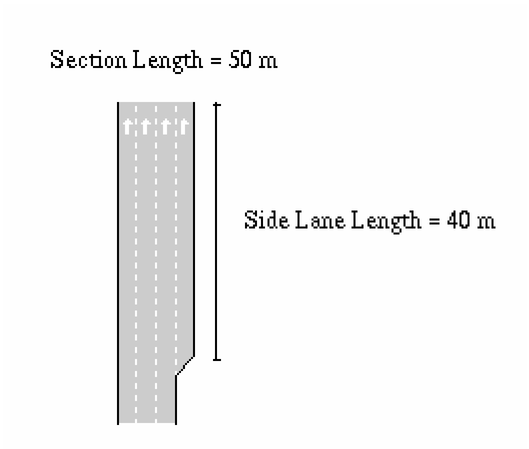


Figure 3.5. Example of a long side lane

Then, the generic capacity per lane in section *s* is defined by

$$GCL_s = C_s \Big/ \sum_{i \in \text{exit lane}} WL_i$$

where C_s is the capacity of section *s*.

The capacity of link *j* (composed by section *s* plus turning movement *t*) is defined by

$$CL_j = \sum_{k \in \text{exit lanes of link j.}} GCL_s * WL_k \; / \; NTS_k$$

where NTS_k is the number of turning movements that lane *k* shares.

Figure 3.6 shows an AIMSUN network composed of three sections (section 1, section 2 and section 3) and an intersection defined by two turning movements: one from section 1 to section 2, as shown in Figure 3.8 (using the two right-most lanes in section 1) and the second, as shown in Figure 3.7, from section 1 to section 3 (using the two left-most lanes in Section 1).

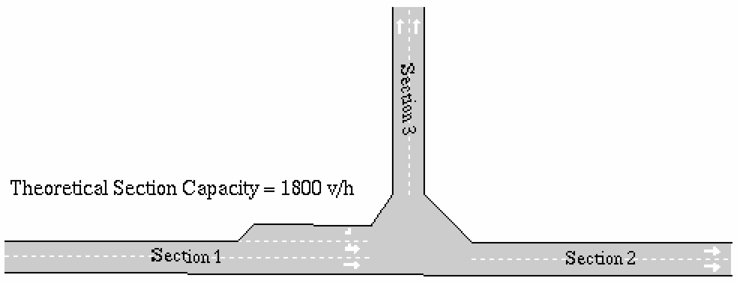


Figure 3.6. Example of the AIMSUN network

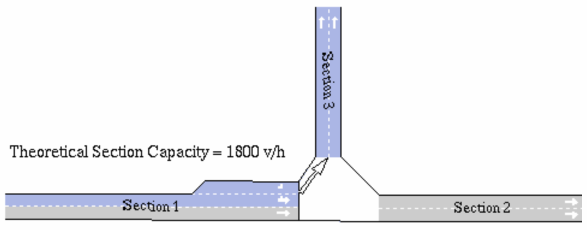


Figure 3.7. Left turning

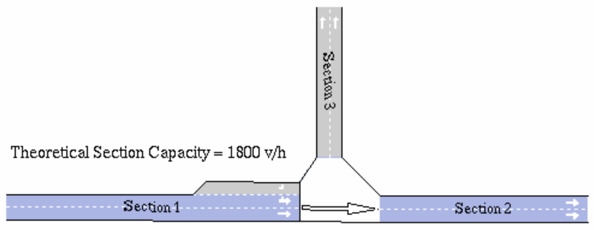


Figure 3.8. Right turning

The corresponding network representation used by the shortest route algorithm, which is composed of nodes and links, is shown in Figure 3.9.

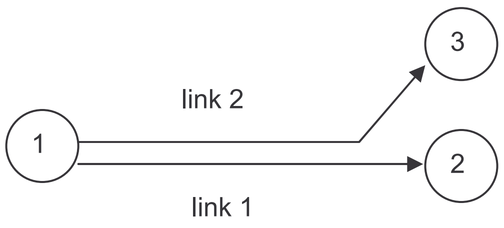


Figure 3.9. Network in Figure 3.6 represented in terms of links and nodes

The generic capacity per lane in section 1, in which the side lane has a weight of 0.35, is

$$GCL_1 = C_1 \Big/ \sum_{i \in \text{exit lane}} WL_1 = 1800/(1+1+0{,}35) = 765{,}96$$

The capacity of link 1 is

$$CL_1 = \sum_{k \in \text{exit lanes of link 1.}} GCL_1 * WL_k \ / \ NTS_k = (765{,}96*1/1) + (765{,}96*1/2) = 1148{,}94$$

and the capacity of link 2 is

$$CL_2 = \sum_{k \in \text{exit lanes of link 2.}} GCL_1 * WL_k \ / \ NTS_k = (765{,}96*1/2) + (765{,}96*0{,}35/1) = 651{,}06$$

3.2.1.2.2 Initial Cost Function

The *initial cost function* is used at the beginning of the simulation when no simulated data have yet been 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. The travel time in free flow conditions is the time it would take a vehicle to cross the link, which is the section plus the turn, assuming that the vehicle is travelling at the maximum speed permitted along the section and at the maximum turning speed during the turning movement. No penalty for traffic lights or traffic signals is considered directly.

There are two types of default initial cost function. For the first type, the vehicle type *IniCost*$_j$ is not considered, that is, all vehicle types have the same initial cost. For the second type, the vehicle type *IniCost*$_{j,\ vt}$ is considered, which means that there is an initial cost function per vehicle type. The presence of reserved lanes in the network determines whether the initial cost function considers vehicle type or not. By default, AIMSUN does not consider vehicle type when choosing the initial cost function, although it does consider vehicle type if there are reserved lanes.

If the penalty for turning *t* that belongs to the link is taken into account, the initial cost of link *j*, *IniCost*$_j$, and its capacity, *CL*$_j$, are calculated as follows:

$$IniCost_j = \mathrm{TravelTFF}_j + \mathrm{TravelTFF}_j * \varphi * (1 - CL_j / \mathrm{C}L_{\max})$$

where

- TravelFF$_\mathrm{j}$ is the estimated travel time of link *j* in free flow conditions.

  $$\mathrm{TravelTFF}_j = \mathrm{Length}_s \big/ \mathrm{SpeedLimit}_s + \mathrm{Length}_t \big/ \mathrm{SpeedLimit}_t$$

  where Length$_s$ and SpeedLimit$_s$ are the length and the speed limit respectively of section *s*, which belongs to link *j*, and Length$_t$ and SpeedLimit$_t$ are the length and the speed limit respectively of the turning *t*, which belongs to link *j*.

- φ is a user-defined capacity weight parameter that allows the user to control the influence of the link capacity on the cost in relation to the travel time.

- *CL*$_{max}$ is the theoretically estimated maximum link capacity in the network.

The initial cost of link *j* per vehicle type *vt*, *IniCost*$_{j,vt}$, is calculated as follows:

$$IniCost_{j,vt} = \mathrm{TravelTFF}_{j,vt} + \mathrm{TravelTFF}_{j,vt} * \varphi * (1 - CL_j / \mathrm{C}L_{\max})$$

where

- TravelFF$_\mathrm{j,vt}$ is the estimated travel time of vehicle type *vt* in link *j* in free flow conditions.

$$\mathrm{TravelTFF}_{j,vt} = \frac{\mathrm{Length}_s}{\mathrm{Min}(\mathrm{SpeedLimit}_s * \theta_{vt}, \mathrm{MaxSpeed}_{vt})} + \frac{\mathrm{Length}_t}{\mathrm{Min}(\mathrm{SpeedLimit}_t * \theta_{vt}, \mathrm{MaxSpeed}_{vt})}$$

where the new parameters, compared with the travel time in free flow conditions regardless of vehicle type, are the average maximum speed of vehicle type *vt, MaxSpeed*$_{vt}$ and the speed acceptance of vehicle type *vt*, θ_{vt}. This parameter (θ

≥ 0) can be interpreted as the 'level of goodness' of the drivers or the degree of acceptance of speed limits. $\theta_{vt} \geq 1$ means that the vehicle will take as its maximum speed for a section a value greater than the speed limit, while $\theta_{vt} \leq 1$ means that the vehicle will use a lower speed limit. φ is a user-defined capacity weight parameter that allows the user to control the influence of the link capacity on the cost in relation to the travel time.

- CL_{max} is the theoretically estimated maximum link capacity in the network.

3.2.1.2.3 Dynamic Cost Function

The *dynamic cost function* is used when there are simulated travel time data available, and therefore it can only be used when the simulation has already started and statistical data have 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. In the first, $DynCost_j$, vehicle type is not considered, which means that all vehicle types have the same cost. In the second, $DynCost_{j,\, vt}$, all vehicle types are considered, which means that there is a dynamic cost function per vehicle type. The presence of reserved lanes in the network determines whether, in the dynamic cost function, vehicle type is considered or not.

The default dynamic cost for link *j* common to all vehicle types, $DynCost_j$, is the mean travel time, in seconds, for all simulated vehicles that have crossed the link during the last data gathering period ($TravelTime_j$). The travel time for link *j* includes the travel time for section *s* plus the travel time for turning movement *t*.

As there may be situations in which no vehicles cross a link, and therefore no travel time information is available, the following algorithm is applied to calculate $EstimatedTravelTime_j$:

```
if ($Flow_j > 0$) then
    $EstimatedTravelTime_j = TravelTime_j$
else
    if (there any vehicle has stopped) then
        $EstimatedTravelTime_j = AvgTimeIn_s$
    else
        $EstimatedTravelTime_j = AverageSectionTravelTime_s$
    endif
endif
$EstimatedTravelTime_j$ = Maximum($EstimatedTravelTime_j$, $TravelFF_j$)
```

According to this algorithm, when several vehicles have crossed link *j* during the last data gathering period ($Flow_j > 0$), the estimated travel time ($EstimatedTravelTime_j$) is taken as the simulated mean travel time. If no vehicle has crossed link *j*, we should distinguish between a totally congested link and an empty link. In the case of congestion, the travel time is calculated as the average waiting time for vehicles that are waiting at the front of the queue in section *s* ($AvgTimeIn_s$). In the case of an empty link, the travel time is taken to be the section travel time, which means that all the turning movements that begin in section *s* are considered ($AverageSectionTravelTime_s$). All travel times calculated must be greater than or equal to the travel time of the link in free flow conditions.

Finally, if the penalty for turning *t* that belongs to the link is taken into account, the dynamic cost of link *j*, $DynCost_j$, and its capacity, CL_j, are calculated as

$$DynCost_j = EstimatedTravelTime_j + EstimatedTravelTime_j * \varphi * (1 - CL_j / \mathrm{CL}_{\max})$$

where

- $EstimatedTravelTime_j$ is the estimated travel time of link *j* calculated using the previous algorithm.
- φ is a user-defined capacity weight parameter that allows the user to control the influence of the link capacity on the cost in relation to the travel time.
- CL_{max} is the theoretically estimated maximum link capacity in the network.

The default dynamic cost for link *j* and vehicle type *vt*, $DynCost_{j,vt}$, is the mean travel time, in seconds, for all simulated vehicles of type *vt* that have crossed the link during the last data gathering period ($TravelTime_{j,vt}$). The travel time for link *j* includes the travel time for section *s* plus the travel time for turning movement *t*.

As there may be situations in which no vehicles of type *vt* cross a link, and therefore no travel time information is available, the following algorithm is applied to calculate $EstimatedTravelTime_{j,vt}$:

```
if ($Flow_{j,vt} > 0$) then
    $EstimatedTravelTime_{j,vt} = TravelTime_{j,vt}$
else
    if (there is any vehicle vt stopped) then
        $EstimatedTravelTime_{j,vt} = AvgTimeIn_{s, vt}$
    else
        if (link j has reserved lanes of vehicle class cl) then
            if (vt belongs to cl) then
                if ($FlowClass_{j, cl} > 0$) then
```

```
                    $EstimatedTravelTime_{j,vt}$ = $TravelTimeClass_{j, cl}$
                else
                    if (there is any vehicle belonging cl stopped) then
                        $EstimatedTravelTime_{j,vt}$ = $AvgTimeInClass_{s, cl}$
                    else
                        $EstimatedTravelTime_{j,vt}$ = $AverageSectionTravelTime_{s, cl}$
                    endif
                endif
            else
                if ($FlowClass_{j, not\ cl}$ > 0) then
                    $EstimatedTravelTime_{j,vt}$ = $TravelTimeClass_{j,not\ cl}$
                else
                    if (there is any vehicle not belonging cl stopped) then
                        $EstimatedTravelTime_{j,vt}$ = $AvgTimeInClass_{s,not\ cl}$
                    else
                        $EstimatedTravelTime_{j,vt}$ = $AverageSectionTravelTime_{s, not\ cl}$
                    endif
                endif
            endif
        else
            if ($Flow_j$ > 0) then
                $EstimatedTravelTime_j$ = $TravelTime_j$
            else
                if (there is any vehicle stopped) then
                    $EstimatedTravelTime_j$ = $AvgTimeIn_s$
                else
                    $EstimatedTravelTime_j$ = $AverageSectionTravelTime_s$
                endif
            endif
        endif
    endif
endif
$EstimatedTravelTime_{j, vt}$ = Maximum($EstimatedTravelTime_{j, vt}$, $TravelFF_{j, vt}$)
```

According to this algorithm, when a vehicle of type *vt* has crossed link *j* during the last data gathering period ($Flow_{j,vt} > 0$), the current cost is taken to be the simulated mean travel time. If no vehicle of type *vt* has crossed link *j*, we distinguish between different cases of costs calculated according to the previous algorithm (each step is carried out if no information is available for the preceding step), where:

- $AvgTimeIn_{s,\,vt}$ is the average waiting time of the first vehicle of vehicle type *vt* at the front of the queue in the section.
- If section *s* has reserved lanes for vehicle class *cl*,
    - and vehicle type *vt* uses the reserved lane:
        - $TravelTimeClass_{j,\,cl}$ is the mean travel time for link *j* when all vehicle types in vehicle class *cl* have been aggregated.
        - $AvgTimeInClass_{s,\,cl}$ is the average waiting time for the first vehicle in vehicle class *cl* at the front of the queue in section *s*.
        - $AverageSectionTravelTime_{s,\,cl}$ is the average section travel time for all vehicles in vehicle class *cl*.
    - and vehicle type *vt* does not use the reserved lane:
        - $TravelTimeClass_{j,\,not\ cl}$ is the mean travel time for link *j* when all vehicle types that do not fall into vehicle class *cl* have been aggregated.
        - $AvgTimeInClass_{s,\,not\ cl}$ is the average waiting time for the first vehicle that does not fall into vehicle class *cl* at the front of the queue in section *s*.
        - $AverageSectionTravelTime_{s,\,not\ cl}$ is the average section travel time for all vehicles that do fall into vehicle class *cl*.
- and section s does not have reserved lanes:
    - $TravelTime_j$ is the mean travel time for link *j* when all vehicle types have been aggregated.
    - $AvgTimeIn_s$ is the average waiting time for the first vehicle at the front of the queue in section *s*.
    - $AverageSectionTravelTime_s$ is the average section travel time for all types.

All calculated travel times must be greater than or equal to the travel time of the link in free flow conditions.

Finally, if the penalty for turning *t* that belongs to the link is taken into account, the dynamic cost of link *j* for vehicle type *vt*, $DynCost_{j,\,vt}$, , and its capacity, CL_j, are calculated as

$$DynCost_{j,vt} = EstimatedTravelTime_{j,vt} + EstimatedTravelTime_{j,vt} * \varphi * (1 - CL_j / \mathrm{C}L_{\max})$$

where

- *EstimatedTravelTime*$_{j,\ vt}$ is the estimated travel time of link *j* for vehicle type *vt* calculated following the previous algorithm.
- φ is a user-defined capacity weight parameter that allows the user to control the influence of the link capacity on the cost in relation to the travel time.
- CL_{max} is the theoretically estimated maximum link capacity in the network.

3.2.1.2.4 USER-DEFINED LINK COST FUNCTIONS

The default cost functions described above are basically defined in terms of link travel time. A wide variety of other link costs are not considered, such as toll pricing, historical travel times that represent drivers' experiences during the preceding days, combinations of the numerical attributes of various links, for instance, travel times, delay times, length and capacity, etc. Therefore, for each particular link, the user may choose to use the initial cost function or the dynamic cost function as the default cost function, or any other cost function defined by the user using the function editor developed in this work, if he or she wishes to take into account other arguments represented by the numerical attributes of other links

User-defined cost functions can be formulated in terms of the most common mathematical functions and operators (+ , -, *, /, ln, log, exp, etc.). Their terms, which can be defined as parameters, constants or variables, must correspond to the numerical attributes of any object in the model (links, sections, turnings, vehicle types, etc.), whose values might be either fixed (i.e. lengths, theoretical capacities, number of lanes, etc.) or subject to change during the simulation (i.e. link flows, average link speeds, average link travel times, etc.). Details of all the cost function types employed in this work may be found in Appendix VI.

3.2.1.3 SHORTEST PATH ALGORITHM

During the simulation, the computation of shortest paths is predefined every predefined time interval Δt. The shortest path routine implemented in this work is a variation of Dijkstra's label-setting algorithm (Dijkstra, 1959), which provides, as its result, the shortest path tree for each destination centroid. This shortest path tree structure therefore provides the shortest path from the start of every section to a given destination. The penalties associated with turning movements are taken into account and thus the cost labels are attached to links rather than nodes, as is usually the case. The link candidate list is stored as a heap data structure. During each iteration of the algorithm, the link that has the minimum cost value is removed from the heap and all the links that are connected backward are added to the heap in the correct position.

The shortest path routine is based on link cost functions, and before it is applied, the costs of all the links are evaluated/updated. At the beginning of the simulation, the initial cost function is evaluated for each link, and during subsequent time intervals, the dynamic cost function, rather than the initial cost function, is evaluated.

The shortest path routine generates a shortest path tree for each destination centroid d (SPT_d, $d \in D$), although there is an additional step that identifies new paths for all OD pairs $i \in I$, taking SPT_d $d \in D$, and adds, to the set of alternatives, path K_i of OD pair i. For one shortest path tree, there are as many SP_{con} paths as the origin centroid has connectors con.

The generic schema for the dynamic traffic assignment is

Step 0. Calculate initial shortest path(s) for each OD pair using the defined initial costs.

Step 1. Simulate for a predefined time interval (e.g. 5 minutes). Assign to the path available the fraction of the trips between each OD pair for that time interval according to the selected route choice model and obtain new average link travel times as a result of the simulation.

Step 2. Recalculate the shortest path, taking into account the experienced average link travel times.

Step 3. If there are guided vehicles or variable message signs that suggest rerouting, provide the drivers who are allowed to dynamically reroute during a trip with the information calculated in Step 2.

Step 4. If all traffic demand has been assigned then STOP Otherwise Go to Step 1.

The algorithm implemented in this work and details of the shortest path calculation are as follows:

```
Step 0. Calculate initial shortest path(s) for each OD pair using the defined initial costs.
    Step 0.1. Initialization:
            Evaluate the initial cost function for each link $j$:
                for each $j \in 1... L$: $Cost_j = InitialCost_j$
    Step 0.2. Apply the shortest path routine:
            for each destination centroid $d$:
                Calculate the shortest path tree $SPT_d$ using $Cost_j$ $j \in 1... L$
    Step 0.3. Identify the shortest path from the shortest path tree:
            for each OD 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 Δt assigning to the available path K_i the fraction of the trips between each OD pair *i* for that time interval according to the selected route choice model.

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

```
Step 2.1. Update link cost functions:
        Evaluate dynamic cost function for each link $j$:
            for each $j \in 1... L$: $Cost_j = DynamicCost_j$
Step 2.2. Apply the shortest path routine:
        for each destination centroid $d$:
            Calculate shortest path tree $SPT_d$ using $Cost_j$ $j \in 1... L$
Step 2.3. Identify the shortest path from the shortest path tree:
        for each OD 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 signs that suggest rerouting, provide the drivers who are allowed to dynamically reroute during a trip with the information calculated in Step 2.

Step 4. If all traffic demand has been assigned then STOP Otherwise Go to Step 1.

3.2.1.4 INITIAL SHORTEST PATH

At the beginning of the simulation, using the initial cost function, a shortest path tree is calculated for each destination centroid, so all vehicles are assigned to the same alternative during the first interval. In order to start considering more than one alternative, as a way of anticipating the assignment process, *k* shortest path trees are calculated at the beginning of the simulation.

The problem of enumerating, in order of increasing length, *k* shortest paths has received considerable attention in the literature (Bellman, 1960), (Dreyfus, 1969), (Fox, 1973), (Eppstein, 1994, 1999). The various algorithms for solving this problem (Eppstein, 1994, 1999), (Jimenez et al., 1999), (Martins, 1984), (Martins et al., 1996) build graphs that represent all the possible deviations from the shortest path, after having computed the shortest path from every node in the graph. Therefore, all *k* shortest paths obtained as a result use the same link length, associated with a cost function which value does not change, while in our case, the calculation of the *k* shortest path takes into account that cost should change depending on the use of link, therefore it must anticipate the evolution of the arc costs whilst considering the traffic flow assigned to each path.

The algorithm implemented in this work calculates, at every iteration, a new shortest path until the number of shortest paths available reaches the parameter *InitialK-SP*. Figure 3.10 shows the generic scheme of the algorithm that iteratively

- evaluates the cost function in each link (at the first iteration, the cost function is the travel time in free flow conditions),
- calculates a new shortest path, and
- determines the path flow (using the incremental loading procedure described below) and updates the flow in each link.

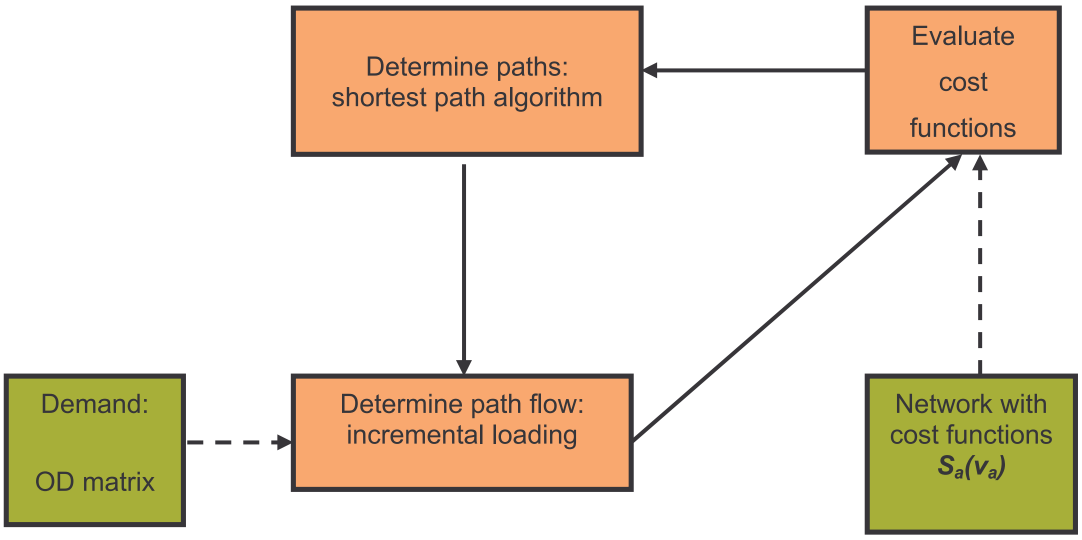

Determine paths: shortest path algorithm
Evaluate cost functions
Demand: OD matrix
Determine path flow: incremental loading
Network with cost functions $S_a(v_a)$

Figure 3.10. Generic scheme of *k* shortest path algorithm

The components of this algorithm are the following:

- Shortest path algorithm. The computation of the shortest path corresponds to a variation of Dijkstra's label-setting algorithm.
- Link cost function. The cost function of link *j,* ($s_j(v_j)$), is a function of the flow in link *j,* v_j, such that

$$s_j(v_j) = tt_0(1+\beta\left[\frac{v_j}{Cj}\right]^{\alpha})$$

  where tt_0 is the travel time in free-flow conditions and C_j the capacity of link *j*.

- Incremental loading algorithm.

  The path flow rates in the feasible region Ω satisfy the conservation flow and non-negativity constraints (where the traffic demand of OD pair *i* is denoted by g_i), such that

$$\Omega = h_k^i : \sum_{k \in K_i} h_k^i = g_i , i \in I; h_k^i \geq 0$$

For each OD pair i $\in$ I and path h_k^i, evaluate the path flow assigned to path $h_k^i \quad k \in K_i$ at iteration *n* $h_k^i(n)$:

$$h_k^i(n) = h_k^i(n-1) + \lambda(g_i - h_k^i(n-1))$$

$$h_l^i(n) = h_l^i(n-1) - \lambda(h_l^i(n-1)), \quad l = 1...(k-1)$$

where $\lambda = {}^{1}\!/_{n+1}$.

The algorithm for calculating 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 OD pair *i* $\in$ *I*, assign $h_k^i(n) = g_i$.

Step 1. Compute the flow v_j for each link *j*

$$v_j = \sum_{i \in I} \sum_{k \in K_i} \delta^i_{j,k} h_k^i$$

where $\delta^i_{j,k} = \begin{cases} 1, \text{link } j \text{ belongs to path } k \text{ of O-D pair } i \\ 0, \text{otherwise} \end{cases}$

Evaluate the cost function of each link *j* ($s_j(v_j)$).

Step 2. *k* = *k* +1, *n* = *n* +1

Compute the *k*-th shortest path-based cost function of each link *j* ($s_j(v_j)$).

Incremental Loading: for each OD pair *i* $\in$ *I*, evaluate

$$h_k^i(n) = h_k^i(n-1) + \lambda(g_i - h_k^i(n-1))$$

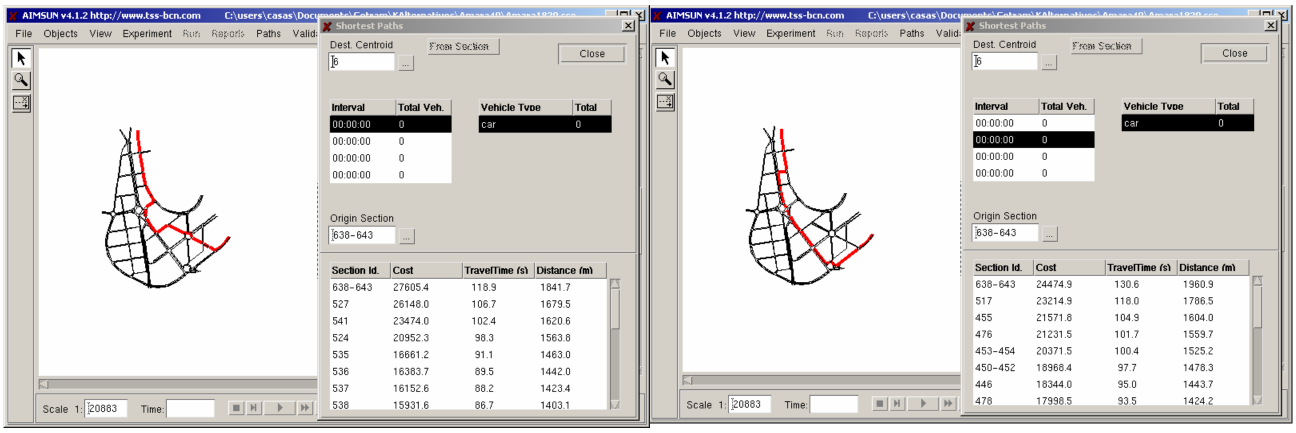
for $l = 1$ to $(k-1)$

$$h_l^i(n) = h_l^i(n-1) - \lambda(h_l^i(n-1)), \quad l = 1...(k-1)$$

where $\lambda = {}^{1}\!/_{n+1}$

Step 3. If *k* is equal to the total number of shortest paths *InitialK-SP,* then stop.

Otherwise, return to Step 1.


AIMSUN v4.1.2 http://www.tss-bcn.com
C:\users\casas\Documents\Getram\KAlternatives\Amara40\Amara1820.sce
File
Objects
View
Experiment
Run
Reports
Paths
Valid
Shortest Paths
From Section
Close
Scale 1:
Time:

Dest. Centroid: 6

Interval	Total Veh.
00:00:00	0
00:00:00	0
00:00:00	0
00:00:00	0

Vehicle Type	Total
car	0

Origin Section: 638-643

Section Id.	Cost	TravelTime (s)	Distance (m)
638-643	27605.4	118.9	1841.7
527	26148.0	106.7	1679.5
541	23474.0	102.4	1620.6
524	20952.3	98.3	1563.8
535	16661.2	91.1	1463.0
536	16383.7	89.5	1442.0
537	16152.6	88.2	1423.4
538	15931.6	86.7	1403.1

Scale 1: 20883

Dest. Centroid: 6

Interval	Total Veh.
00:00:00	0
00:00:00	0
00:00:00	0
00:00:00	0

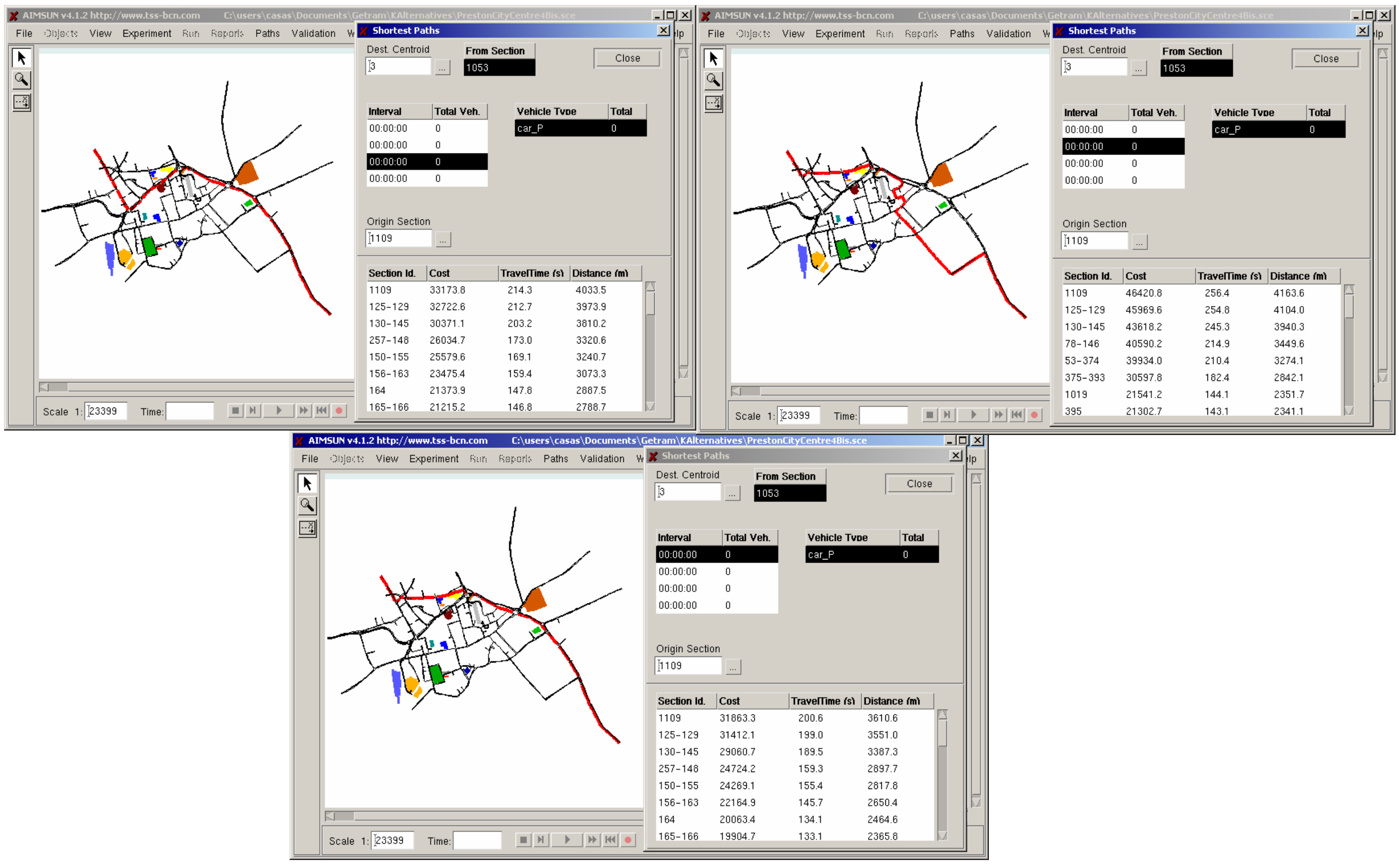
Vehicle Type	Total
car	0

Origin Section: 638-643

Section Id.	Cost	TravelTime (s)	Distance (m)
638-643	24474.9	130.6	1960.9
517	23214.9	118.0	1786.5
455	21571.8	104.9	1604.0
476	21231.5	101.7	1559.7
453-454	20371.5	100.4	1525.2
450-452	18968.4	97.7	1478.3
446	18344.0	95.0	1443.7
478	17998.5	93.5	1424.2

Scale 1: 20883

Figure 3.11. Example of the *k* shortest path in the Amara model


AIMSUN v4.1.2 http://www.tss-bcn.com
C:\users\casas\Documents\Getram\KAlternatives\PrestonCityCentre4Bis.sce
File
Objects
View
Experiment
Run
Reports
Paths
Validation
W
Help
Shortest Paths
Close
Scale 1:
Time:

Dest. Centroid: 3

From Section
1053

Interval	Total Veh.
00:00:00	0
00:00:00	0
00:00:00	0
00:00:00	0

Vehicle Type	Total
car_P	0

Origin Section: 1109

Section Id.	Cost	TravelTime (s)	Distance (m)
1109	33173.8	214.3	4033.5
125-129	32722.6	212.7	3973.9
130-145	30371.1	203.2	3810.2
257-148	26034.7	173.0	3320.6
150-155	25579.6	169.1	3240.7
156-163	23475.4	159.4	3073.3
164	21373.9	147.8	2887.5
165-166	21215.2	146.8	2788.7

Scale 1: 23399

Dest. Centroid: 3

From Section
1053

Interval	Total Veh.
00:00:00	0
00:00:00	0
00:00:00	0
00:00:00	0

Vehicle Type	Total
car_P	0

Origin Section: 1109

Section Id.	Cost	TravelTime (s)	Distance (m)
1109	46420.8	256.4	4163.6
125-129	45969.6	254.8	4104.0
130-145	43618.2	245.3	3940.3
78-146	40590.2	214.9	3449.6
53-374	39934.0	210.4	3274.1
375-393	30597.8	182.4	2842.1
1019	21541.2	144.1	2351.7
395	21302.7	143.1	2341.1

Scale 1: 23399

Dest. Centroid: 3

From Section
1053

Interval	Total Veh.
00:00:00	0
00:00:00	0
00:00:00	0
00:00:00	0

Vehicle Type	Total
car_P	0

Origin Section: 1109

Section Id.	Cost	TravelTime (s)	Distance (m)
1109	31863.3	200.6	3610.6
125-129	31412.1	199.0	3551.0
130-145	29060.7	189.5	3387.3
257-148	24724.2	159.3	2897.7
150-155	24269.1	155.4	2817.8
156-163	22164.9	145.7	2650.4
164	20063.4	134.1	2464.6
165-166	19904.7	133.1	2365.8

Scale 1: 23399

Figure 3.12. Example of the *k* shortest path in the Preston model

The preceding figures depict examples of two and three initial *k* shortest paths, respectively. The first example, Figure 3.11, is a medium-sized model of the borough of Amara in the city of San Sebastian in Spain, and the second, Figure 3.12, represents a partial model of Preston in the UK.

3.2.2 PATH SELECTION

The path selection process, which is based on discrete route choice models, involves estimating path flow rates and modelling the driver's choice of path from a given set of alternatives that connect an origin to a destination. The decision is made when a vehicle enters the system (initial assignment) and during its trip, when new alternatives become available (en-route assignment).

Given a set of alternative paths, the probability of each available path is calculated during the path selection process and then the driver's decision is modelled by randomly selecting an alternative path according to the probabilities assigned to each alternative. This process corresponds to Step 1 in the generic dynamic assignment algorithm and thus the algorithm is

```
Step 0.   Calculate initial shortest path(s) for each OD pair using the initial costs
          defined.
    Step 0.1. Initialization:
              Evaluate initial cost function for each link $j$:
                  for each $j \in 1... L$: $Cost_j = InitialCost_j$
    Step 0.2. Apply shortest path routine:
              for each destination centroid $d$:
                  Calculate shortest path tree $SPT_d$ using $Cost_j$ $j \in 1... L$
    Step 0.3. Identify shortest path from shortest path tree:
              for each OD 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 $\Delta t$ and assign the fraction of the trips
          between each OD pair $i$ for that time interval to the available path $K_i$, according
          to the selected route choice model.
    Step 1.1. Assignment of path probabilities:
                  for each OD pair $i$:
                      Calculate $P_k$ using the route choice model, where $k \in K_i$
    Step 1.2. Simulate for a predefined time interval $\Delta t$, generate the fraction of the
              vehicles between each OD pair $i$ for that time interval, and randomly
              select the path according to probabilities $P_{k}$, $k \in K_i$.

Step 2.   Recalculate the shortest path, taking into account the average link travel times
          experimented.
    Step 2.1. Update the link cost functions:
            Evaluate dynamic cost function for each link $j$:
                for each $j \in 1... L$: $Cost_j = DynamicCost_j$
    Step 2.2. Apply the shortest path routine:
            for each destination centroid $d$:
                Calculate shortest path tree $SPT_d$ using $Cost_j$ $j \in 1... L$
    Step 2.3. Identify the shortest path from shortest path tree:
            for each OD 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 signs that suggest rerouting,
          provide the drivers who are allowed to dynamically reroute during a trip with
          the information calculated in Step 2.
Step 4.   If all traffic demand has been assigned then STOP Otherwise Go to Step 1.
```

Candidate paths can be of two different types (this is explained in more detail in Section 3.2.1): user-defined paths (UDP) and shortest paths calculated, which can be calculated using the initial cost function, *initial shortest paths* (ISP) or the dynamic cost function, *dynamic shortest paths* (DSP).

A vehicle of type *vt* travelling from OD pair *i* can choose one path according to the user-defined assignment or as a result of a route choice model (explained in section 3.2.2.1 and section 3.2.2.2) from the set of alternative paths K_i:

- N user-defined paths: UdP_n^i $n=1..N$
- M initial shortest paths: ISP_m^i , $m=1..M$
- P timely updated shortest paths: DSP_p^i, $p=1..P$

3.2.2.1 USER-DEFINED ASSIGNMENT

User-defined assignment means that, for each user-defined path, the user determines the probability of usage and distinguishes between vehicle types. The user defines the probability of use of user-defined paths and the probability of use of the initial shortest path

$P(UdP_n^i, vt)$: Probability of use UdP_n^i by vehicle type *vt*

$P(ISP_m^i, vt)$: Probability of use ISP_m^i by vehicle type *vt*

satisfying the following condition:

$$\sum_{n=1}^{N} P(UdP_n^i, vt) + \sum_{m=1}^{M} P(ISP_m^i, vt) \leq 1 \quad ,i \in I$$

3.2.2.2 Route choice Models

At any time during the simulation, there will be a set of alternative paths for each OD pair. The driver's decision to select one of the available paths, that is, to assign a trip to a path, can be emulated with a route choice model. Route choice models are usually inspired by the discrete choice theory that determines the probability of choosing an alternative from a set of alternatives as a function of its utility. From the point of view of transport, the most common value associated with a trip is the travel time or travel cost, which represents a disutility. Therefore, route choice models should be formulated in terms of this negative utility. The most common concept of path cost assumes that is additive, so the cost of path *i* CP_i is computed as the sum of the costs of the links $Cost_j$ (as explained above), and the path is composed as follows:

$$CP_i = \sum_{link\ j \in path\ i} Cost_j$$

The default route-choice models that are available are proportional, multinomial logit and C-logit, but the user can also define his or her own user-defined route choice model using the function editor.

3.2.2.2.1 Proportional

The choice probability P_k of a given alternative path *k*, where $k \in K_i$, can be expressed as

$$P_k = \frac{CP_k^{-\alpha}}{\sum_{l \in K_i} CP_l^{-\alpha}}$$

where CP_i is the cost of path *i*.

When $\alpha = 1$, the probability is inversely proportional to the path cost. The example in Table 3.1 illustrates how the alpha factor (α) influences the probability of choosing a given path in the case of there being two alternative paths for which the travel times are 5 and 4 minutes respectively.

Path Costs	Path 1	Path 2	
	300	240	seconds
	0,08333333	0,06666667	hours
Alpha	**P(1)**	**P(2)**	
0	0.5	0.5	

0.2	0.4888447	0.5111553
0.4	0.4777004	0.5222996
0.6	0.4665784	0.5334216
0.8	0.4554894	0.5445106
1	0.4444444	0.5555556
1.2	0.4334541	0.5665459
1.4	0.4225288	0.5774712
1.6	0.4116788	0.5883212
1.8	0.4009140	0.5990860
2	0.3902439	0.6097561
2.2	0.3796778	0.6203222
2.4	0.3692246	0.6307754
2.6	0.3588927	0.6411073
2.8	0.3486901	0.6513099
3	0.3386243	0.6613757
3.2	0.3287025	0.6712975
3.4	0.3189312	0.6810688
3.6	0.3093165	0.6906835
3.8	0.2998640	0.7001360
4	0.2905789	0.7094211

Table 3.1. Alpha factor in the proportional route choice model

Figure 3.13 depicts the role of the alpha factor as a function of the different path costs. The alpha factor α is a user-defined parameter, so it can be used to adjust the effects that small changes in the travel times may have on the driver's decisions.

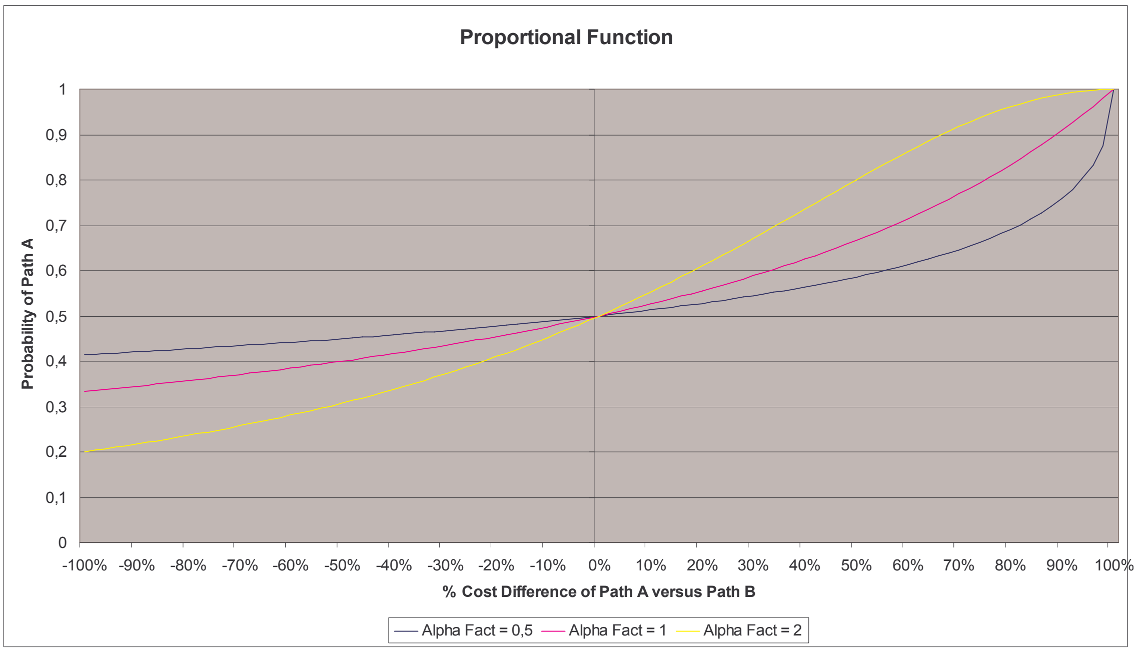


Figure 3.13. Proportional function shape

The estimation of the most appropriate value for the α factor from real data is context-dependent and therefore so is part of the calibration process, as described in detail in Section 5.2.

3.2.2.2.2 Multinomial Logit

The choice probability P_k of a given alternative path k, where $k \in K_i$, can be expressed as a function of the difference between the measured utilities of that path and all other alternative paths, such that

$$P_k = \frac{e^{v_k \theta}}{\sum_{l \in K_i} e^{v_l \theta}}$$

or its equivalent expression

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

where V_i is the perceived utility of alternative path i and θ is a shape or scale factor. We have assumed $V_i = -CP_i/3600$ (function V_i, from which the cost of path i has been subtracted, is measured in hours).

We assume that the utility U_k^i of path k between OD pair i is given by

$$U_k^i = -\theta t_k^i + \varepsilon_k^i$$

where

θ is a shape or scale factor parameter

t_k^i is the expected travel time on path k of OD pair i, and

ε_k^i is a random term.

The underlying modelling hypothesis is that random terms ε_k^i are independent identically distributed Gumbel variates[1]. Under these conditions, the probability of choosing path *k* from amongst all alternative routes of OD pair *i* is given by the following logistic distribution:

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

The role of the scale factor θ is two-fold: it makes the decision based on differences between utilities and independently of measurement units, and it influences the standard error of the distribution of expected travel times, such that

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

that is,

$\theta < 1$ means that there is a high perception of the variance or, in other words, a trend towards using many alternative routes.

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

For example, given four alternative paths for which the expected travel times (cost paths) are T1 = 12 minutes, T_2 = 15 minutes, T_3 = 16 minutes and T_4 =18 minutes, the corresponding probabilities according to Equation (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 $P_4 = 0.03535$. Figure 3.14 depicts the role of the scale factor, as a function of the difference between path costs.

[1] A random variable ε is Gumbel-distributed if $F(\varepsilon) = \exp\left[-e^{-\mu(\varepsilon-\eta)}\right]$, where $\mu>0$ and η is a location parameter. If $\varepsilon_i, \varepsilon_j$ are independent Gumbel variates with parameters (η_1, μ) and (η_2, μ) respectively, then $\varepsilon^* = \varepsilon_i - \varepsilon_j$ is logistically distributed, i.e. $F(\varepsilon) = \left[1+e^{\mu(\eta_2-\eta_1-\varepsilon^*)}\right]^{-1}$ and if they are identically distributed, i.e. $(\eta_1 = \eta_2)$, then $F(\varepsilon) = \left[1+e^{-\mu\varepsilon^*}\right]^{-1}$.

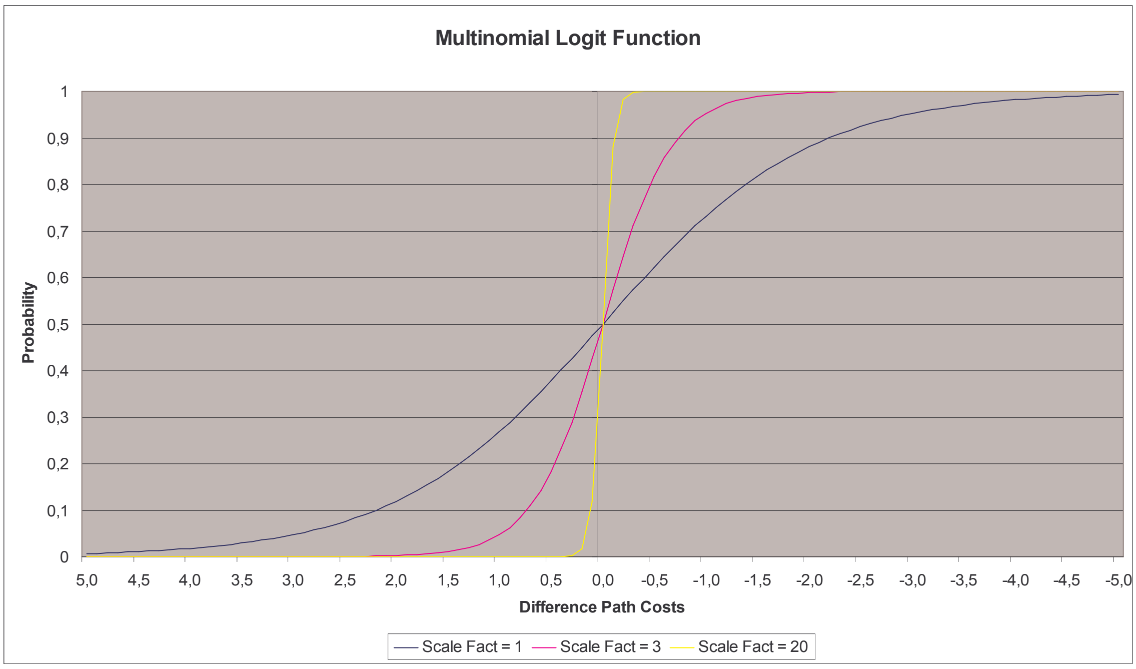


Figure 3.14. Logit function shape

The two-fold role of θ can also be explained by decomposing it and formulating the logit model as

$$P_k = \frac{e^{\mu V_k}}{\sum_j e^{\mu V_j}}$$

so when $V_k = -\beta t_k$ (t_k being the travel time for path k), $\theta = \mu\beta$ captures the sensitivity to time (β) and the sensitivity of the scale (μ) to the utility. Our case, however, usually requires more complex cost functions, as discussed above.

The parameter or scale factor θ is a user-defined parameter, so it can be used to adjust the effects that small changes in the travel times may have on the driver's decisions. The example in Table 3.2 illustrates how the scale factor (θ) influences the probability of choosing a given path if there are two alternative paths for which the travel times are 5 minutes and 4 minutes respectively.

Path Cost	**Path 1**	**Path 2**	
	300	240	Seconds
	5	4	Minutes
	0.083333	0.066667	Hours
Scale Factor	**P(1)**	**P(2)**	
1	0.495833	0.504167	
10	0.458430	0.541570	

20	0.417430	0.582570
30	0.377541	0.622459
40	0.339244	0.660756
50	0.302941	0.697059
60	0.268941	0.731059
100	0.158869	0.841131
500	0.000240	0.999760
1000	0	1
2000	0	1
3600	0	1

Table 3.2. Multinomial logit example

3.2.2.2.3 C-LOGIT

The logit models exhibit a tendency towards route oscillations in the routes used, and the corresponding instability generates a kind of flip-flop process. According to our experience, there are two main reasons for this behaviour: the properties of the logit function and the inability of the logit function to distinguish between two alternative routes when there is a high degree of overlapping. Several researchers, such as Cascetta (1996), have recently reported these drawbacks.

The instability of the routes used can be substantially improved if the network topology allows for alternative paths with little or no overlapping at all, by changing the shape factor θ and recomputing paths very frequently. However, in large networks in which there are many alternative paths between origins and destinations, some of which overlap to a certain extent (see Figure 3.15), the use of the logit function may still exhibit some weaknesses.

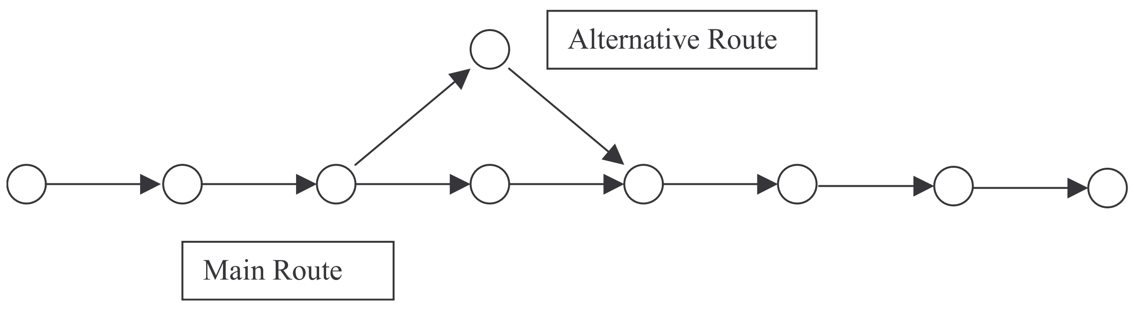


Figure 3.15. Overlapping paths

To overcome these weaknesses, the C-logit model (Cascetta, 1996) was implemented. In the C-logit model, which is, in fact, a variation of the logit model, the choice probability P_k, of each alternative path k belonging to the set $\in K_i$ of available paths that connect an OD pair is expressed as

$$P_k = \frac{e^{\theta(V_k - CF_k)}}{\sum_{l \in K_i} e^{\theta(V_l - CF_l)}}$$

where V_i is the perceived utility of alternative path *i* and θ is the scale factor, as in the case of the logit model.

The ‘commonality factor’ of path *k*, denoted as CF_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 a smaller utility with respect to similar paths. CF_k is calculated as follows:

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

where L_{lk} is the length of the links common to paths *l* and *k*, while L_l and L_k are the length of paths *l* and *k* respectively. Depending on the two factor parameters β and γ, a greater or lesser weighting is given to the ‘commonality factor’. Larger values of β mean that the overlapping factor is of greater significance with respect to the utility *Vi*; γ is a positive parameter, which is usually taken to be in the range [0, 2]. The latter’s influence is less than that of β and it has the opposite effect.

As a rule of thumb, one may suggest taking factor β to be in the range $[t_{min}, t_{max}]$, where $t_{min} = \mathrm{Min}_{k \in Ki} [CP_k]$ and $t_{max} = \mathrm{Max}_{k \in Ki} [CP_k]$. Thus, β becomes a kind of scaling factor for CF_k, which translates it into an order of magnitude similar to that of V_k in the formula $V_k - C_k$ that is used for the exponential. Furthermore, when using larger values of β, it is possible that the ‘commonality factor’ CF_k will have a greater influence on the choice probability than the utility (i.e. the travel time) itself, which would lead to a higher probability of choosing longer, non-overlapping paths than shorter, heavily overlapping paths.

The following is an example that illustrates the use of C-logit and compares it to the multinomial logit. Figure 3.16 shows a hypothetical network that has four alternative paths between origin O and destination D. Table 3.3 represents the resulting choice probabilities for both models, where the overlapping(*l/k*) between path *l* and path *k* is the sum of cost of all common links in hours.

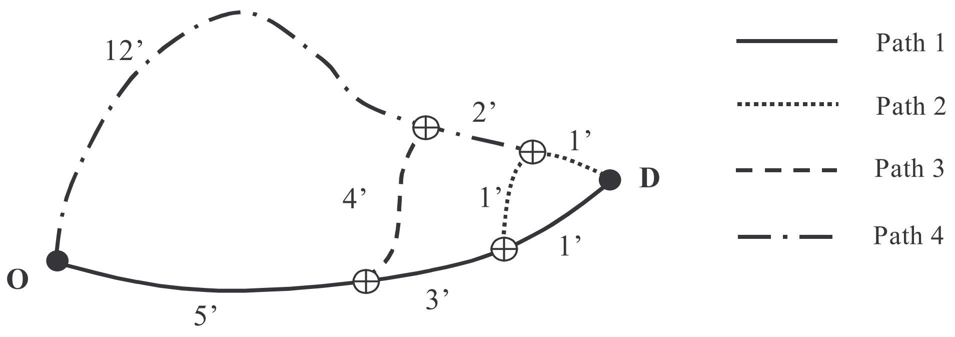


Figure 3.16. Example of network that has overlapped paths

Travel Times	**Path 1**	**Path 2**	**Path 3**	**Path 4**		
	540	600	720	900	seconds	
	9	10	12	15	minutes	
	0.15	0.166666667	0.2	0.25	hours	
Overlapping (l*lk*)						
l / k	**1**	**2**	**3**	**4**		
1	0.1500	0.1333	0.0833	0.0000		
2	0.1333	0.1667	0.1000	0.0167		
3	0.0833	0.1000	0.2000	0.0500		
4	0.0000	0.0167	0.0500	0.2500		
					Beta	**0.15**
CFk	0.007029	0.007544	0.006767	0.002220	**Gamma**	**1**

Example of LOGIT

Scale Factor	**P(1)**	**P(2)**	**P(3)**	**P(4)**
1	0.260448	0.256143	0.247746	0.235663
10	0.354498	0.300076	0.215014	0.130412
20	0.450502	0.322799	0.165730	0.060969
30	0.532071	0.322717	0.118721	0.026490
40	0.599856	0.307976	0.081182	0.010987
50	0.656417	0.285278	0.053882	0.004423
60	0.704153	0.259044	0.035058	0.001745
100	0.836359	0.157968	0.005635	0.000038
500	0.999760	0.000240	0.000000	0.000000
1000	1.000000	0.000000	0.000000	0.000000
2000	1.000000	0.000000	0.000000	0.000000
3600	1.000000	0.000000	0.000000	0.000000

Example of C-LOGIT

Scale Factor	**P(1)**	**P(2)**	**P(3)**	**P(4)**
1	0.253734	0.247897	0.246238	0.252131
10	0.288035	0.228222	0.213405	0.270338
20	0.327050	0.205324	0.179528	0.288098
30	0.366177	0.182150	0.148925	0.302747
40	0.404621	0.159478	0.121923	0.313978
50	0.441725	0.137948	0.098616	0.321711
60	0.477006	0.118032	0.078900	0.326062
100	0.596019	0.058128	0.029706	0.316146
500	0.959694	0.000008	0.000000	0.040297
1000	0.998240	0.000000	0.000000	0.001760
2000	0.999997	0.000000	0.000000	0.000003
3600	1.000000	0.000000	0.000000	0.000000

Table 3.3. Comparison between logit and C-logit

3.2.2.2.4 User-defined Route Choice Model

As an alternative to the default route choice models, the user can define his or her own route choice models using the function editor implemented in this work.

The user-defined route choice model has to give the probability for each one of *n* alternative paths to be chosen by vehicles, whilst taking into account vehicle type.

User-defined cost functions can be formulated in terms of the most common mathematical functions and operators (+, -, *, /, ln, log, exp, ∑, ∏, etc.). Their terms, which can be defined in terms of parameters, constants and variables, must correspond to the numerical attributes of any object in the model (paths, vehicle types, etc.), whose values might be either fixed (i.e. lengths, theoretical capacities, desired speed, etc.) or subject to change during the simulation (i.e. path flows, path costs, path travel times, etc.). Appendix VI should be referred to for details of the expression of a user-defined route choice function.

3.2.2.3 Determining the set of paths in the decision-making process

The path selection process, which is based on discrete route choice models, involves estimating the path flow rates of a set of alternative paths K_i that connect the OD pair *i*-th. This set of alternative paths is updated considering the shortest path trees calculated at every interval in the dynamic traffic assignment.

An aspect that must be studied is the impact on the dynamic traffic assignment results of two different approaches to determining the set of alternative paths K_i that connect the OD pair *i*-th at time interval *t*.

At time interval *t*, the shortest path algorithm is applied and it generates the shortest path tree $SPT_d(t)$ for each destination centroid $d \in D$. Then, for each OD pair $i \in I$ (whose origin centroid is *o* and whose destination centroid is *d*), it takes $SPT_d(t)$ and generates as many paths $SP_{con}(t)$ as the origin centroid has connectors *con*.

- A first approach receives *MaxNumberSPT* as a parameter (which represents the last number of the shortest path tree to be considered). The algorithm is the following:

```
for each OD pair i (from origin centroid o to destination d):
    $K_i = \emptyset$
    for $j \in 0 \dots (MaxNumberSPT-1)$:
        for $con \in$ Connectors of origin centroid o:
            Add $SP_{con}(t-j)$ to $K_i$
            endfor
        endfor
    endfor
```

- A second approach receives *MaxNumberSPT* as a parameter (which represents the last number of the shortest path tree to be considered) and *MaxNumberRoutes* (which represents the maximum number of different paths). The algorithm is the following:

```
for each OD pair $i$ (from origin centroid $o$ to destination $d$):
    $K_i = \emptyset$
    for $con \in$ Connectors of origin centroid $o$:
        $K_i(con) = \emptyset$
        for $j \in 0 \dots (MaxNumberSPT-1)$:
            if $SP_{con}(t-j) \notin K_i$
                Add $SP_{con}(t-j)$ to $K_i$
            endif
        endfor
        Order $K_i(con)$ by cost ascendant: cost of
        for $m \in 1 \dots MaxNumberRoutes$:
            Add element $m$ of $K_i(con)$ to $K_i$
        endfor
    endfor
endfor
```

3.2.2.4 INITIAL ASSIGNMENT

The initial assignment is the process of selecting a path when a vehicle enters the system, which corresponds to Step 1 in the generic dynamic traffic assignment algorithm (see Section 3.2)

A vehicle of type *vt* travelling from OD pair *i* may choose one path according to the user-defined assignment or, as a result of a route choice model, from the set of alternative paths K_i:

- *N* user-defined paths: UdP_n^i $n = 1..N$
- *M* initial shortest paths: ISP_m^i , $m = 1..M$
- *P* timely updated shortest paths: DSP_p^i, $p = 1..P$

The initial assignment algorithm has two parts depending on whether vehicle *v* is guided or not.

- If vehicle *v* is guided (an attribute of the vehicle), it directly assigns the probability calculated using the route choice model to all the alternative paths k, $k \in K_i$.
- If a vehicle *v* is unguided and the sum of probabilities $PUdA = \sum_{n=1}^{N} P(UdP_n^i, vt) + \sum_{m=1}^{M} P(ISP_m^i, vt)$ is equal to 1.0, it assigns the probabilities defined by means of the user-defined assignment to the user-defined paths. If the aforementioned sum is less than 1.0, then *PUdA* will represent the probability of the vehicle using the user-defined assignment and the rest (1.0 - *PUdA*) will apply the route choice model and assign the probability calculated using the route choice model to all the alternative paths *k*, $k \in K_i$.

The vehicle is then randomly assigned to one path according to all the path probabilities assigned.

The following algorithm defines the initial assignment path of vehicle *v*, which is of type *vt* and belongs to OD pair *i* (from origin *Oi* to destination *Dj*).

```
if vehicle v is guided then
    Apply route choice model:
        Generate random number X inside the interval defined between [0...1]
        Accumulated = 0
        for each k, $k \in K_i$:
            if X <= Accumulated + $P_k$ then
                Assign path k to vehicle v and $\boxed{\text{STOP}}$
            else
                Accumulated = Accumulated + $P_k$
            endif
else
    Generate random number X inside the interval defined between [0...1]
```

```
if $\sum_{n=1}^{N} P(UdP_n^i, vt) + \sum_{m=1}^{M} P(ISP_m^i, vt)$ <= $X$ then
    Apply user-defined assignment:
        Accumulated = 0
        for each $UdP_n^i$, $n \in 1...N$:
            if $X$ <= Accumulated + $P(UdP_n^i, vt)$ then
                Assign path $UdP_n^i$ to vehicle $v$ and $\boxed{\text{STOP}}$
            else
                Accumulated = Accumulated + $P(UdP_n^i, vt)$
            endif
        for each $ISP_m^i$, $m \in 1...M$:
            if $X$ <= Accumulated + $P(ISP_m^i, vt)$ then
                Assign path $ISP_m^i$ to vehicle $v$ and $\boxed{\text{STOP}}$
            else
                Accumulated = Accumulated + $P(ISP_m^i, vt)$
            endif
else
    Apply route choice model:
        Generate random number $X$ defined between [0...1]
        Accumulated = 0
        for each $k$, $k \in K_i$:
            if $X$ <= Accumulated + $P_k$ then
                Assign path $k$ to vehicle $v$ and $\boxed{\text{STOP}}$
            else
                Accumulated = Accumulated + $P_k$
            endif
    endif
endif
```

3.2.2.5 EN-ROUTE ASSIGNMENT

Vehicles are initially assigned to a path from a set of available paths based on probability. Apart from the initial assignment, which is made at the time of the vehicle's departure, there is the possibility of making a path reassignment during the trip (en-route assignment).

The en-route assignment corresponds to Step 3 in the generic dynamic traffic assignment algorithm (see Section 3.2).

A guided vehicle can make a new decision about which path to follow at any time during a trip whenever new shortest paths become available. The algorithm is defined as

```
for all vehicle $v$ guided then
    Generate random number $X$ defined between [0...1]
    Accumulated = 0
    for each $k$, $k \in K_i$:
        if $X$ <= Accumulated + $P_k$ then
            Assign path $k$ to vehicle $v$ and $\boxed{\text{STOP}}$
        else
            Accumulated = Accumulated + $P_k$
        endif
```

3.2.2.6 ENTRANCE/EXIT SECTION DECISION

When the route choice model is applied to a vehicle in the initial assignment, the first step is to select the entrance section at which the vehicle will enter the system and the exit section at which it will exit. This decision is made by considering the use of percentages in the origin and destination centroids (attribute of centroid, see Appendix V for details). There are therefore the following four different scenarios:

a) Origin centroid considers percentages/destination centroid considers percentages:

The example in Figure 3.1 demonstrates the decision-making process, in which the origin centroid has 3 connections (A, B and C) of 30%, 20% and 50% respectively, and the destination centroid has 3 exit connections (X, Y and Z) of 40%, 10% and 50%. The only feasible combinations that the connectivity of the network allows are that, from connection B, it is possible to reach connections X, Y and Z, while from connection C it is possible to reach connections Y and Z. All other combinations are unfeasible.

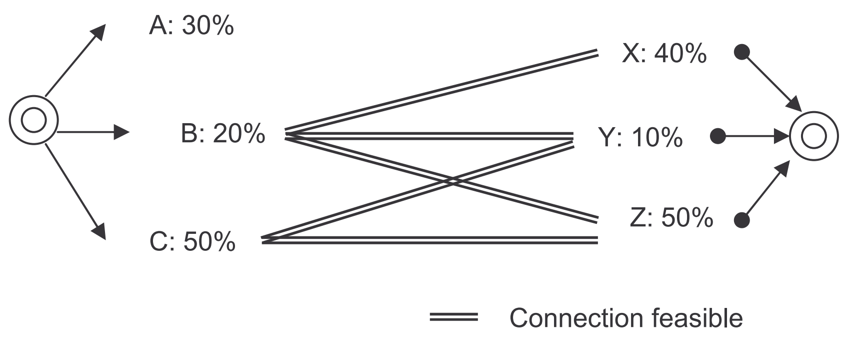


Figure 3.17. Example of origin/considers percentages

The first step is to choose the entrance section. This process takes into account only the feasible connections for reaching the destination and recalculates the percentages by taking the original percentage over the sum of percentages of all feasible connections. In our example, the percentage of A is 0% (unfeasible), the percentage of

B is 20%/70% (70% is the sum of 20% and 50%) and the percentage of C is 50%/70%. On the basis of the percentages calculated, the vehicle chooses the connector (i.e. the entrance section) randomly.

Once the entrance section has been determined, it is necessary to determine the exit section, by considering only the feasible connections from the entrance section and recalculating the percentages (the original percentage over the sum of percentages of all feasible connections). In our example, if the entrance section chosen uses connection C, then the probability of connection X will be 0% (unfeasible from connection C), that of Y will be 10%/60% (60% is the sum of 10% and 50%) and that of Z will be 50%/60%. The vehicle then chooses the exit section randomly.

b) Origin centroid considers percentages/destination centroid does not consider percentages

In this case, the entrance section is selected as in a) and the exit section depends on the shortest path.

c) Origin centroid does not consider percentages/destination centroid considers percentages

In this case, the exit section is selected as in case a), by considering only the feasible connectors from the origin. The entrance section is selected by considering the shortest path.

d) Origin centroid does not consider percentages/destination centroid does not consider percentages

The entrance section and the exit section are selected by considering the shortest path.

3.3 Reactive Dynamic Traffic Assignment

3.3.1 Preventive and Reactive Dynamic Traffic Assignment

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

- A method for determining the path-dependent flow rates on the paths in the network.
- A dynamic network loading method, which determines how these path flows give rise to time-dependent arc volumes, arc travel times and path travel times.

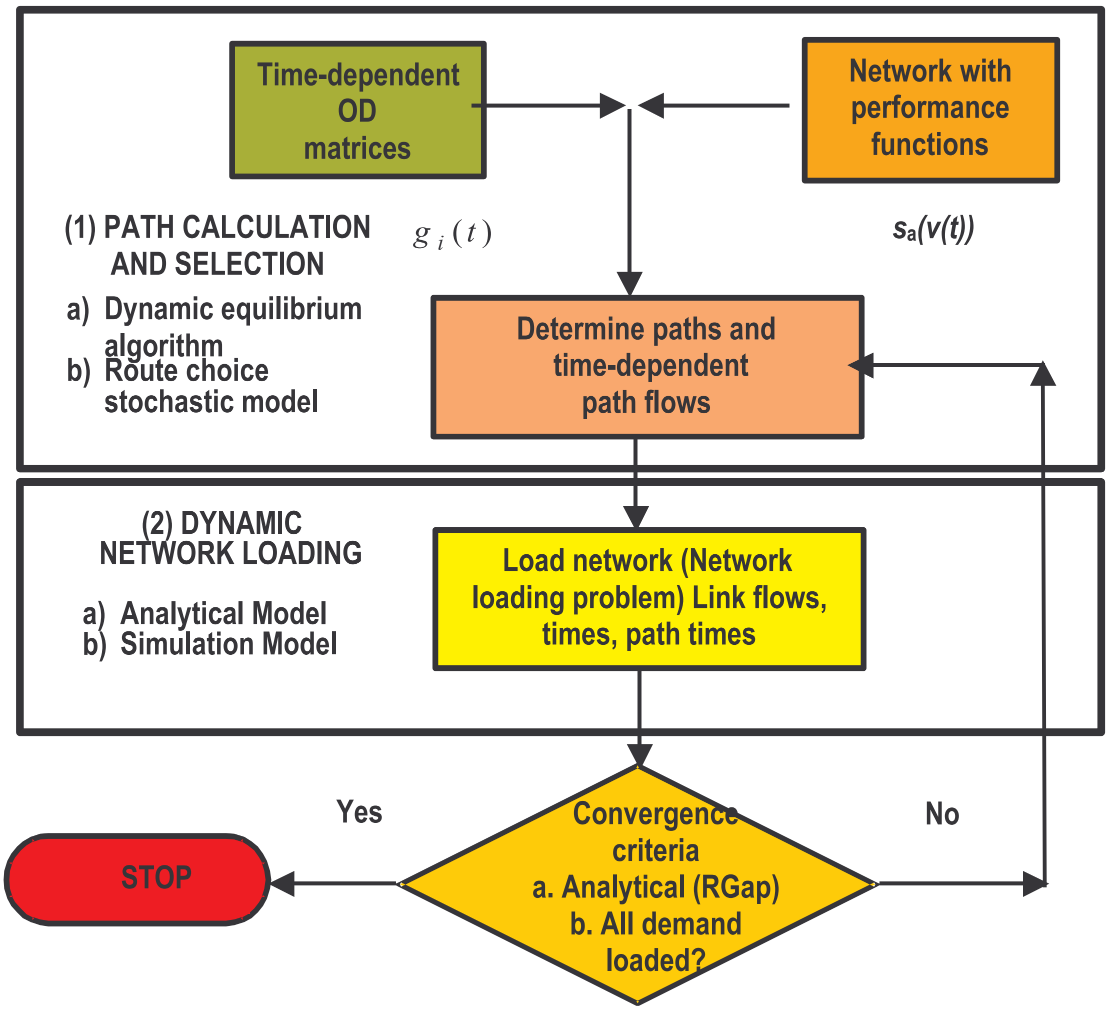

Time-dependent OD matrices
Network with performance functions
(1) PATH CALCULATION AND SELECTION
a) Dynamic equilibrium algorithm
b) Route choice stochastic model
$g_i(t)$
$s_a(v(t))$
Determine paths and time-dependent path flows
(2) DYNAMIC NETWORK LOADING
a) Analytical Model
b) Simulation Model
Load network (Network loading problem) Link flows, times, path times
Yes
No
STOP
Convergence criteria
a. Analytical (RGap)
b. All demand loaded?

Figure 3.18. Conceptual approach to dynamic traffic assignment

The diagram in Figure 3.18 depicts the logical, conceptual approach to dynamic traffic assignment models. Path flow rates depend on the emulation of drivers' path choice behaviour. Two alternative approaches may be considered:

- Preventive dynamic assignment or dynamic en-route assignment. At each time period, the corresponding fraction of the demand is assigned to the currently available paths for each origin-destination pair according to the probabilities estimated by a route choice model. Drivers can be allowed to dynamically change route en route if a better path from their current position to their destination becomes available.

- Reactive dynamic assignment or dynamic equilibrium assignment. Path flows are determined by an approximate solution to the mathematical model for dynamic equilibrium conditions (Florian et al., 2001), (Barceló et al., 2002).

The simulation process based on time-dependent paths comprises the following steps:

Step 0. Calculate initial shortest path(s) for each OD pair using the defined initial costs.

```
Step 0.1. Initialization:
        Evaluate the initial cost function for each link $j$:
            for each $j \in 1 \ldots L$: $Cost_j = InitialCost_j$
Step 0.2. Apply the shortest path routine:
        for each destination centroid $d$:
            Calculate shortest path tree $SPT_d$ using $Cost_j \; j \in 1 \ldots L$
Step 0.3. Identify the shortest path from the shortest path tree:
        for each OD 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 Δt (e.g. 5 minutes) assigning to the available path K_i the fraction of the trips between each OD pair *i*-th for that time interval according to the probabilities $P_k \; k \in K_i$ estimated by the selected route choice model and obtain new average link travel times as a result of the simulation.

Step 2. Update the link cost functions and recalculate the shortest path, taking into account the experimented average link travel times.

```
Step 2.1. Update the link cost functions:
        Evaluate the dynamic cost function for each link $j$:
            for each $j \in 1 \ldots L$: $Cost_j = DynamicCost_j$
Step 2.2. Apply the shortest path routine:
        for each destination centroid $d$:
            Calculate the shortest path tree $SPT_d$ using $Cost_j \; j \in 1 \ldots L$
Step 2.3. Identify the shortest path from the shortest path tree:
        for each OD 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 signs that suggest rerouting, provide the drivers who are allowed to dynamically reroute during a trip with the information calculated in Step 2.

Step 4. Preventive dynamic assignment:

If all the demand has been assigned, then stop.

Otherwise, go to Step 1.

Reactive dynamic assignment:

If all the demand has been assigned and the convergence criteria hold, then stop.

Otherwise,

Go to Step 1 if all the demand has not yet been assigned , or

Go to Step 0 and start a new major iteration.

3.3.2 REACTIVE DYNAMIC TRAFFIC ASSIGNMENT IN AIMSUN

In the new approach, implemented in AIMSUN following the dynamic equilibrium assignment or reactive dynamic assignment, the simulation is replicated *N* times and link costs for each link *j*, for each time interval *t, t+1, ..., L* (where $L = T/\Delta t$, *T* being the simulation horizon and Δt the user-defined time interval in which to update paths and path flows) and every iteration *n* are stored. Thus at iteration *n* the link costs of previous iteration *n-1* can be used in an anticipatory day-to-day learning mechanism to estimate the expected link cost at the current iteration. Figure 3.19 depicts this day-to-day learning mechanism, where at the first iteration (*n* = 0) the simulation receives as input the expected link costs initialised as the expected travel time. Free flow conditions are assumed and the experimented link costs are generated as simulation output. At iteration *n*, the simulation model receives as input the expected link cost calculated using the link costs at iteration *n*-1 (a combination of the expected and experimented link costs). This iterative process ends when a convergence criterion (either a maximum number of iterations or the user-equilibrium criterion) is reached.

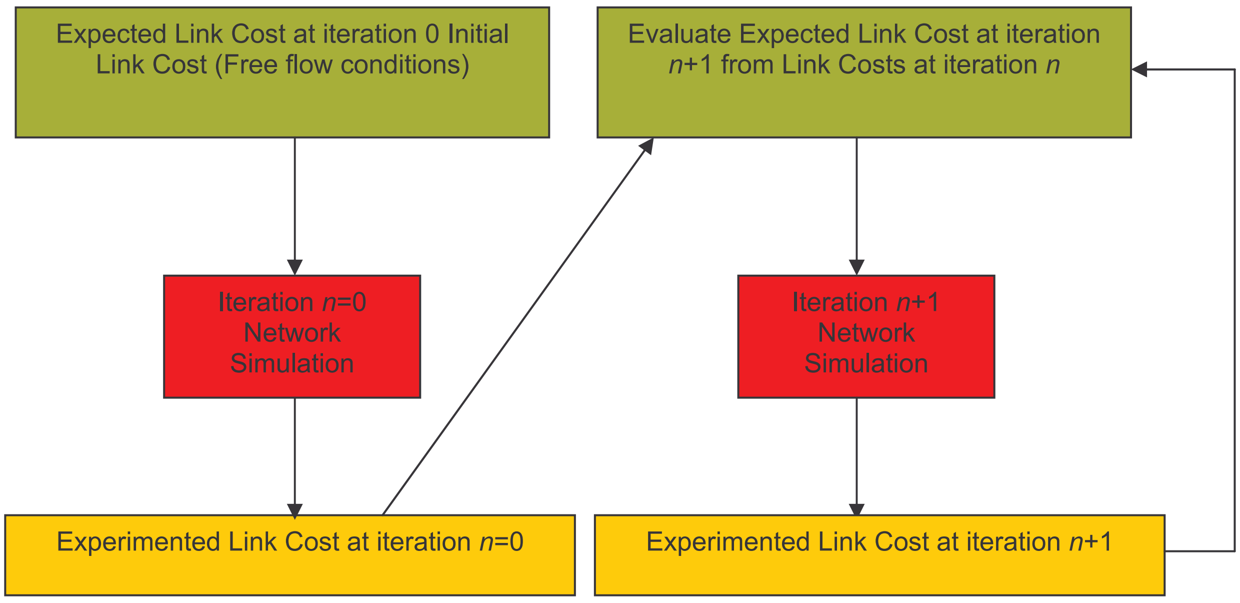

Expected Link Cost at iteration 0 Initial Link Cost (Free flow conditions)
Evaluate Expected Link Cost at iteration n+1 from Link Costs at iteration n
Iteration n=0 Network Simulation
Iteration n+1 Network Simulation
Experimented Link Cost at iteration n=0
Experimented Link Cost at iteration n+1

Figure 3.19. Scheme of day-to-day learning mechanism

The following algorithm represents the modified dynamic traffic assignment in AIMSUN (where the modifications introduced are represented in bold):

```
Step a: n = 0

Step b: Simulation Model

    Step 0.    Calculate initial shortest path(s) for each OD pair using the defined initial costs
        Step 0.1. Initialization:
                Evaluate Initial Cost Function per each link $j$:
                    for each $j \in 1 \ldots L$: $Cost_j = InitialCost_j$
        Step 0.2. Apply Shortest Path routine:
                for each destination centroid $d$:
                    Calculate Shortest Path Tree $SPT_d$ using $Cost_j\ j \in 1 \ldots L$
        Step 0.3. Identify Shortest Path from Shortest Path Tree:
                for each OD pair $i$ (from origin centroid $o$ to destination $d$)
                    Add to Path(s) $SP_{con}$ to $K_i$
        Step 0.4. Read Expected and Experimented Link Cost of Replication n-1
               if n = 0 then Expected and Experimented Link Cost is the travel time
               in free flow conditions otherwise is read from Replication n-1
    Step 1.    Simulate for a predefined time interval $\Delta t$ assigning to the available path $K_i$ the
               fraction of the trips between each OD pair $i$ for that time interval according to
               the selected route choice model.
    Step 2.    Recalculate shortest path, taking into account the experimented average link
               travel times.
        Step 2.1. Update Link Cost Functions:
               Store Cost Link per each link $j$ as Experimented Cost at iteration n at
               time interval t-1.
                Evaluate dynamic cost function per each link $j$:
                    for each $j \in 1 \ldots L$ : $Cost_j$ is a combination of the expected and
                    experimented cost of link $j$
               Store cost link per each link $j$ as expected cost at iteration $k$ at time
               interval $t$.
        Step 2.2. Apply shortest path routine:
                for each destination centroid $d$:
                    Calculate shortest path tree $SPT_d$ using $Cost_j\ j \in 1 \ldots L$
        Step 2.3. Identify shortest path from shortest path tree:
                for each OD 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 signs that suggest rerouting, provide the drivers who are allowed to dynamically reroute during a trip with the information calculated in Step 2.

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

Step c. Update Iteration counter *n* = *n* + 1

Step d: If the Convergence Criteria is not fulfilled, go to Step b.

The main point in this algorithm is to evaluate the dynamic link cost function considering the experimented and expected link cost, either by taking previous intervals from the same iteration (the same simulation) or taking the experimented and expected travel time from previous iterations.

In the subsequent section, the new function is analysed as dynamic link cost function.

3.3.2.1 LINK COST FUNCTION: COMBINATION OF THE EXPECTED AND EXPERIMENTED LINK COST OF THE PREVIOUS ITERATION

Let $C_j^n(t)$ be the input cost of link *j* at iteration *n* at time interval *t,* and let $\hat{C}_j^n(t)$ be the output cost of link *j* at iteration *n* at time interval *t*. The input cost $C_j^n(t)$ could be interpreted as the expected cost considered at interval *t*, while the output cost $\hat{C}_j^n(t)$ could be interpreted as the experimented or experienced link cost at the end of interval *t*.

The link cost function defined in this new approach takes the link cost to be a convex combination of the input cost of link *j* at iteration *n* at time interval *t* and the output cost of link *j* at iteration *n* at time interval *t*.

$$C_j^{n+1}(t) = \lambda\, C_j^n(t) + (1-\lambda)\ \hat{C}_j^n(t) \text{, where } 0 \le \lambda \le 1$$

Figure 3.20 illustrates how the link cost function is evaluated for each iteration and each time interval.

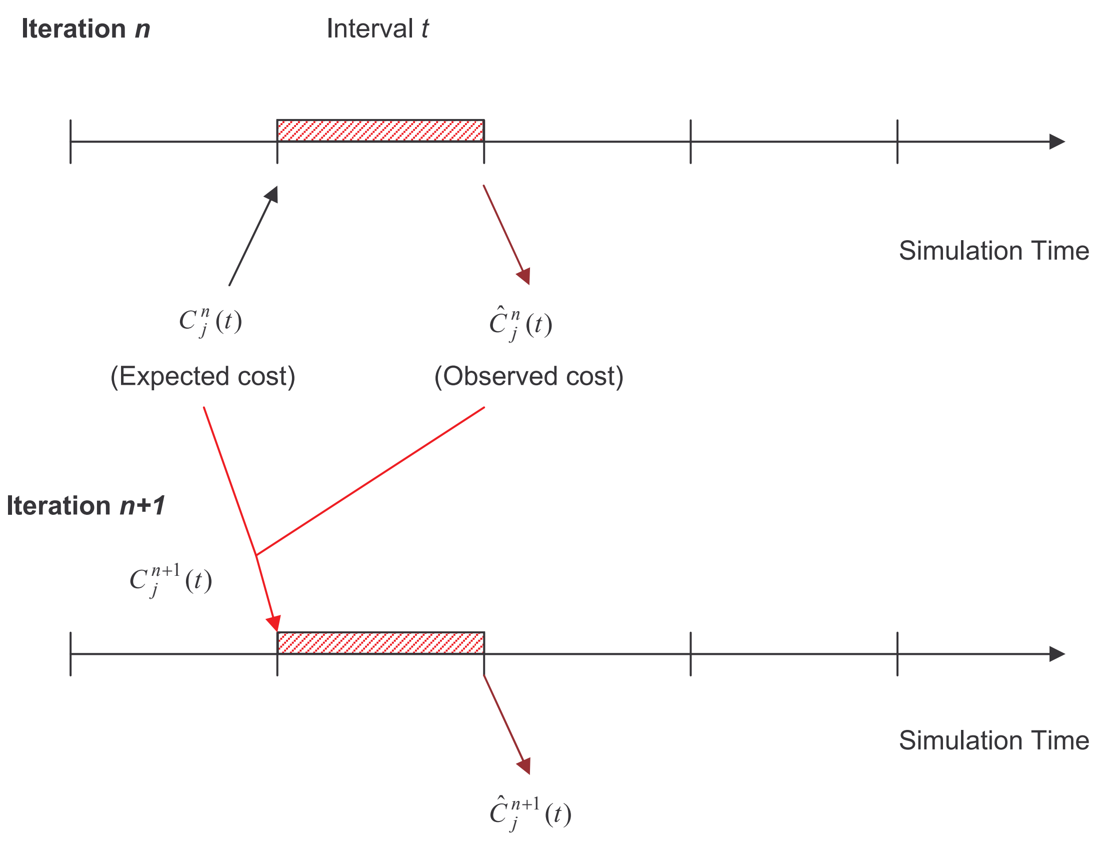


Figure 3.20. Scheme of link cost function evaluation

The resulting cost of path *k* at time interval *t* is

$$\tilde{S}_k(h^{n+1}, t) = \sum_{j \in A} C_j^{n+1}(t)\delta_{jk}$$

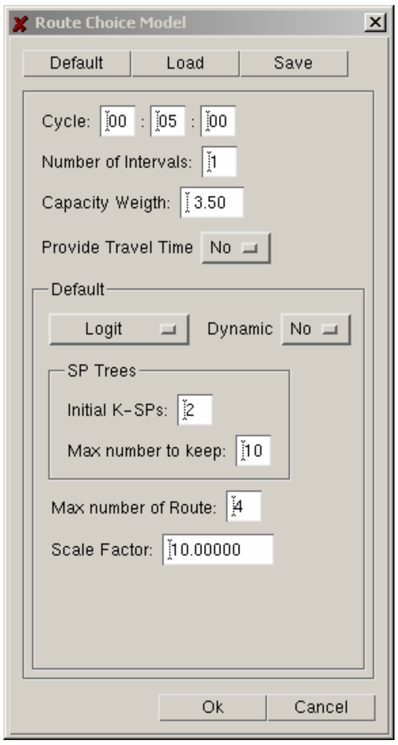
where, as before, δ_{jk} is 1 if link *j* belongs to path *k* and 0 otherwise. Path costs $\tilde{S}_k(h^{n+1}, t)$ are the arguments of the route choice function (logit, C-logit, user-defined, etc.) used at iteration *n+1* to split the demand g_i^{n+1} between the paths available for OD pair *i*.

3.4 Dynamic Traffic Assignment Parameters

The dynamic traffic assignment parameters are as follows (Figure 3.21 depicts the dialog that enables the user to define all the parameters involved in the dynamic traffic assignment algorithm):

- *Cycle*. This parameter defines the time interval Δt used in the dynamic traffic assignment algorithm (Step 1, see Section 3.2) and it defines the interval between recalculations of the shortest path. This time interval is used to evaluate the *RGap* function.

- *Number of Intervals*. When the dynamic cost function of each link is evaluated, it uses the data observed during the simulation (i.e. link travel time). This observed data is taken from the last number of intervals, where the length of this interval is determined by the cycle parameter. Figure 3.22 depicts an example in which the cycle is 5 minutes and the number of intervals is 2. At time 18:20, the link cost evaluation uses the data observed from 18:10 to 18:20.



Cycle: 00 : 05 : 00
Number of Intervals: 1
Capacity Weigth: 3.50
Provide Travel Time: No

Default
Logit
Dynamic: No

SP Trees
Initial K-SPs: 2
Max number to keep: 10

Max number of Route: 4
Scale Factor: 10.00000

Figure 3.21. Dialog for defining the dynamic traffic assignment parameters

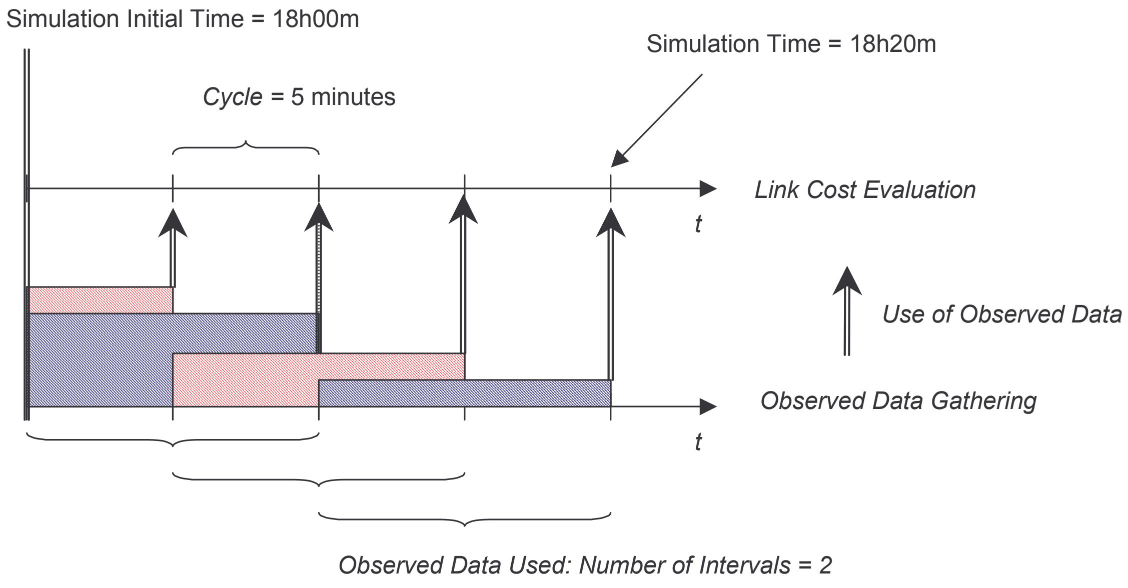


Figure 3.22. Use of observed data in link cost functions

- *Capacity Weight*. This is a user-defined capacity weight parameter that allows the user to control the influence that the link capacity has on the cost in relation to the travel time (see Section 3.2.1.2)

- *Route choice model*. This parameter determines the discrete choice model used during the simulation. The user can choose between proportional, logit, C-logit or user-defined route choice models (see Section 3.2.2.2 for details).

- *Initial K-SP.* This parameter defines the number of shortest paths calculated at the beginning of the simulation. If this parameter is equal to 1, only one shortest path tree is calculated per destination centroid at the beginning of the simulation, taking into account the initial cost function (see Section 3.2.1.2.2). Otherwise, it calculates the *k* shortest path tree using the algorithm shown in Section 3.2.1.4.

- *Max number to keep (MaxNumberSPT).* This parameter defines the maximum number of shortest path trees per destination centroid that are kept in the memory during the simulation. One step in the dynamic traffic assignment algorithm is to identify the set of alternatives path K_i for each OD pair i and the maximum number to keep parameter determines the number of the last shortest path tree that must be considered in order to generate all alternative paths of K_i .The aim of this parameter is to save memory during the simulation and to improve the performance. If this parameter multiplied by the cycle parameter is greater than the simulation time, then all the shortest path trees calculated are kept during the entire simulation.

- *Max number of routes.* This parameter defines the maximum number of different paths used in the path selection process (see Section 5.1).

- Several additional parameters are related to particular route choice models (see Section 3.2.2.2 for details).

    - Proportional: alpha factor.

    - Logit: scale factor.

    - C-logit: scale factor, beta and gamma.

3.5 PATH ANALYSIS TOOL

Appendix VI gives details (such as the graphical user interface developed and the output database) of the implementation of the path analysis tool in AIMSUN, which enabled the research in this thesis to be carried out.

3.5.1 Path Analysis

To gain insight of what is happening in a heuristic dynamic assignment for the proper calibration and validation of the simulation model, the user should have access to the analysis of the routes used. The path information that is available is as follows:

- Shortest path information. The user can view all the shortest path information that is being used by vehicles during the simulation.
- User-defined path information. The user can view all the user-defined path information.
- Shortest path display. The user can simultaneously view different shortest paths and their links in the network.
- Initial path assignment. The user can view all the probabilities considered when a vehicle enters the system.
- Dynamic path assignment. The user can view all the probabilities considered when a vehicle carries out path reassignment during the trip.

3.5.2 Simulation Output

The simulation output for validating the dynamic traffic assignment is the following: the user-defined paths displayed, the shortest path calculated and the initial and en-route path assignment displayed. These capabilities are complemented by generating the path information output, generating link cost information and colouring all the vehicles per origin, per destination and both (per origin and destination simultaneously).